NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

Label Noise Injection Methods for Model Robustness Assessment in Fraud Detection Datasets

Sofia Jerónimo dos Santos

Internship Report presented as the partial requirement for obtaining a Master's degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

Label Noise Injection Methods for Model Robustness Assessment in Fraud Detection Datasets

by

Sofia Jerónimo dos Santos

Internship Report presented as the partial requirement for obtaining a Master's degree in Data Science and Advanced Analytics

Advisor: Mauro Castelli

Co Advisor: Maria Inês Pastor Pereira da Silva, João Guilherme Simões Bravo Ferreira

August 2020

Acknowledgements

First and foremost, I would like to express my deep gratitude to my two supervisors and researchers at Feedzai, Maria Silva and João Bravo for all the guidance, support and fun times throughout the project. I also want to thank Ricardo Barata for all the comments and discussion which were essential for the improvement of this report. I am also thankful to Feedzai for giving me the opportunity to do my report in the research team and be so open to new ideas.

To my colleagues and teachers, and to everyone that in one way or another contributed to my personal and academic growth, a huge thank you.

I would like to thank João Lourenço and Bruno Candeias, for developing the NovaIMS latex template.

Finally to my family, boyfriend, and friends for always supporting me.

Abstract

Label noise is a common issue in real-life applications of machine learning for fraud detection, that can lead to sub-optimal decisions during the model building phase, and, ultimately, lead to poor model performance. A key factor to the impact of noisy data on the performance of a model is the algorithm used to train and its robustness to label noise. In this work, we studied the robustness of the models generated by two different supervised tree-based algorithms, Random Forest and LightGBM, to different types of random and not at random artificial label noise injection techniques, at different percentages of noise, and using different datasets to both train and evaluate them. We also observed the impacts of label noise in the evaluation of the performance of a model. Finally, we analyzed the importance of the different hyperparameters of both algorithms in their performance. We show that both algorithms are robust to random label noise at different noise percentages, however they fail to separate between the classes when in the presence of noise not at random. We also show that, for random label noise, the correlation between the model performance over the noisy validation set and the test set decreases as we increase the noise percentage, however, for noise not at random there is no obvious correlation between the two sets. Finally, we conclude which hyperparameters are the most relevant for the performance of Random Forest models in the presence of random label noise, and in most cases, neither of the studied hyperparameters for LightGBM seem to be more relevant than the others for model performance.

Keywords Label noise Fraud detection Random Forest LightGBM Model robustness Hyperparameter importance

Resumo

Um problema comum na aplicação de técnicas de aprendizagem automática para a deteção de fraude é a rotulagem incorreta das instâncias, que pode levar a decisões sub-ótimas durante a fase de construção do modelo, e assim levar a que o mesmo tenha baixo desempenho. Um fator-chave do impacto que a rotulagem incorreta tem no desempenho de um modelo é o algoritmo usado na sua construção e o quão robusto é. Neste trabalho, estudámos a robustez de modelos gerados através de dois tipos diferentes de algoritmos de aprendizagem supervisionado baseados em árvores de decisão, Random Forest e LightGBM, a diferentes tipos de métodos de injeção de ruído, uns aleatórios e outros determinísticos. Avaliámos os resultados adicionando diferentes percentagens de perturbação no treino e na validação e analisámos o impacto do ruído tanto no treino, como na avaliação do desempenho do modelo. Por fim, analisámos a importância dos diferentes hiper-parâmetros têm para o aumento do nível de desempenho do modelo. Os nossos resultados mostram que ambos os algoritmos são robustos a diferentes percentagens de rótulos incorretos, quando estes são introduzidos de forma aleatória, contudo os algoritmos não conseguem distinguir entre casos de fraude e de não fraude quando são usados métodos determinísticos. Vamos também mostrar que, para rótulos incorretos introduzidos de forma aleatória, a correlação entre o desempenho de um modelo nos dados de validação com ruído e o desempenho do modelo nos dados de teste sem ruído, diminui à medida que aumentamos a percentagem de rótulos incorretos. Porém, para métodos determinísticos de inserção de rótulos incorretos, não se verifica nenhuma correlação entre os conjuntos de dados. Concluímos quais os hiper-parâmetros que são mais relevantes para o desempenho dos modelos de Random Forest quando consideramos a inserção aleatória de rótulos incorretos, e que para LightGBM, na maior parte das vezes, nenhum dos hiper-parâmetros estudados se parece destacar quando consideramos o desempenho do modelo.

Palavras-Chave Rótulos Incorretos Deteção de Fraude Random Forest LightGBM Robustez Importância dos Hiper-parâmetros

Contents

List of Figures **xiii**

List of Tables **xv**

Acronyms **xvii**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Company history . . . 4
1.3 Problem and contribution . . . 4

2 Literature review **7**
2.1 Noise in classification problems . . . 7
2.2 Artificial label noise injection methods . . . 8
2.3 Impacts of label noise . . . 10
2.4 Machine Learning modeling . . . 10
2.4.1 Tree-based supervised models . . . 11
2.4.2 Robustness to label noise . . . 13
2.4.3 Evaluation . . . 15
2.4.4 Hyperparameter search and importance . . . 16

3 Methodology **19**
3.1 Fraud detection datasets . . . 19
3.1.1 Data preprocessing . . . 20
3.1.2 Data split . . . 20
3.2 Types of artificial label noise . . . 21
3.2.1 Injection of artificial label noise . . . 22
3.3 Modeling . . . 23
3.3.1 Model training and hyperparameter tuning . . . 23
3.3.2 Model evaluation . . . 24
3.4 Programming language and packages . . . 26

4 Results and discussion **27**

4.1 Model robustness 27
4.2 Evaluation on noisy validation 33
4.3 Hyperparameter importance 35

5 Conclusions and Future Work **43**
5.1 Conclusions 43
5.2 Future Work 44

Bibliography **47**

Appendices **53**

A Appendix 1 **53**

List of Figures

1.1 Life cycle of a transaction. Image taken from Feedzai documentation. . . 2

2.1 Example of the labels that were disturbed in NNAR NL and NNAR NN injection methods 9

3.1 Experiment setup. 22

4.1 Model performance in the test set per each noise type. 28
4.2 ELpAUC measure for RF and LGBM models in presence of not at random label noise. 30
4.3 RF probability distribution for the selected best model in the noisy validation when in presence of NCAR, NAR1, NNAR NL and NNAR NN in CC test set. 31
4.4 LGBM probability distribution for the selected best model in the noisy validation when in presence of NCAR, NAR1, NNAR NL and NNAR NN in CC test set. In the case of random label noises, each boxplot represents one random seed. 32
4.5 Correlation between the noisy validation and the test pAUC for NCAR. . 34
4.6 RF's pAUC score for the 100 sets of hyperparameters tested. 37
4.7 Variance contribution in the test set for the RF parameters in NCAR and NAR0. 37
4.8 RF's pAUC score for the different values of CCP alpha in NCAR scenario. 38
4.9 LGBM's pAUC score for the 100 sets of hyperparameters tested in NCAR and NAR scenarios. 39
4.10 LightGBM's pAUC score for the different values of minimum sum hessian in leaf in NAR1 scenario in CC dataset. 41

List of Tables

2.1 Impact of label noise and parameter robustness in tree-based models. . . 14

3.1 Datasets considered for the artificial label noise experiment. 20
3.2 Time span and number of transactions for train, validation and test set for each dataset. 21
3.3 Hyperparameter values and description used in RF and LGBM models. . 25

4.1 Best model performance and robustness in the unchanged test set for Random Forest (RF) and LightGBM (LGBM) when the models were tuned and selected in the noisy validation set. 29
4.2 Pearson correlation between original or noisy validation set and the test set. 33
4.3 Variance of performance per model and dataset and noise type 36

A.1 SVM performance results for nonlinearwise label noise. 53
A.2 Characteristics of the artificial label noise in the training set. 53

Acronyms

AUC	Area Under the Curve
Banksim	Synthetic Data from a Financial Payment System
CC	Credit Card Fraud Detection
CCP	Cost-Complexity Pruning
DT	Decision Tree
EFB	Exclusive Feature Bundling
ELA	Equalized Loss Accuracy
ELpAUC	Equalized Loss pAUC
fANOVA	Functional ANOVA
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GBM	Gradient Boosting Machine
GOSS	Gradient-based One-side Sampling
IEEE	IEEE-CIS Fraud Detection
IG	Information Gain
KPI	Key Performance Indicators
LDA	Linear Discriminant Analysis
LGBM	LightGBM

MDI	Mean Decrease Impurity
NAR	Noise at Random
NAR0	Noise at Random in the negative class
NAR1	Noise at Random in the positive class
NCAR	Noise Completely at Random
NNAR	Noise Not at Random
NNAR NL	Noise Not at Random Non Linearwise
NNAR NN	Noise Not at Random Neighbourwise
pAUC	partial AUC
PCA	Principal Component Analysis
RF	Random Forest
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TPR	True Positive Rate
XGBoost	Extreme Gradient Boosting

Chapter 1

Introduction

1.1 Motivation

Nowadays, using debit or credit cards to make purchases is a part of our daily lives as it is very simple and convenient to use. However, transactions are processed by a very complex system which involves multiple entities. A cardholder, the owner of the debit or credit card issued by a bank. A merchant, which represents any business engaged in the sale of goods or services. An acquiring bank, which is a registered member of the card associations that allows merchants to accept card payments and deposits the funds into the merchant's bank account. A card issuer, also known as the cardholder's bank, which is the entity that issues the cardholder's card and pays the acquiring bank for purchases that the cardholder did. Finally, a Card Network, the entity that provides the communication system between the card issuer and the acquiring bank.

When a cardholder uses a debit or credit card to buy a product or service from a merchant, it starts a flow of information that transmits the data about the transaction from the merchant to the acquirer, card network, card issuer and ending in the cardholder. After the information is processed, a flow of financial funds is then started by the cardholder that pays the total amount of the transaction to the card issuer, which then will compensate the credit card network, that in its turn compensates the acquirer. Finally, the acquirer will compensate the merchant, while keeping its commission. Figure 1.1 illustrates the life cycle of a transaction, as we described it.

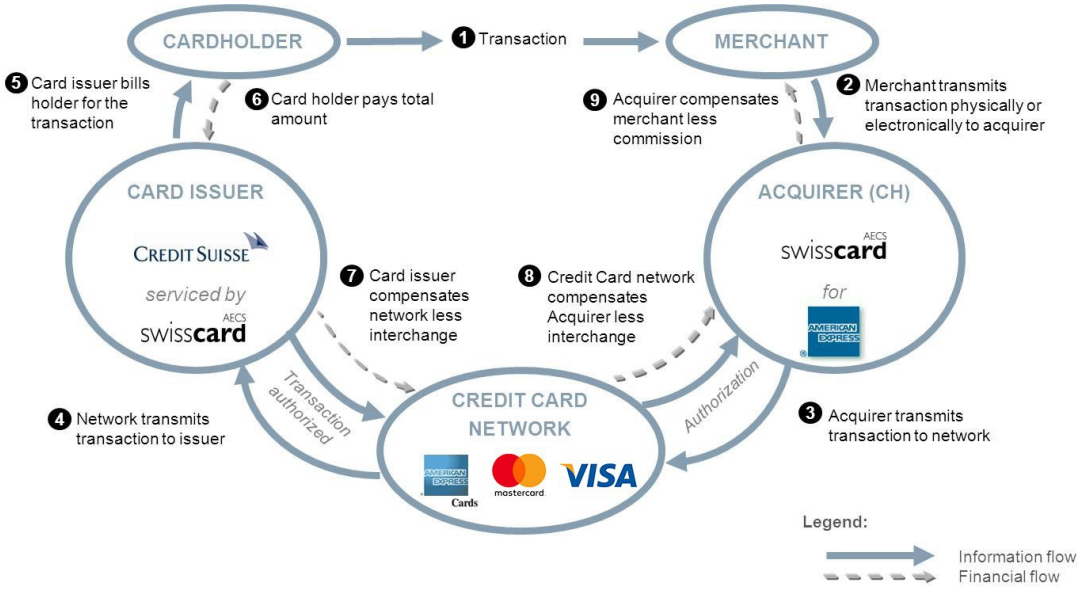


Figure 1.1: Life cycle of a transaction. Image taken from Feedzai documentation.

Every day, billions of transactions between merchants and cardholders are performed worldwide which are in its great majority legitimate transactions. However, criminals keep finding ways of committing fraud. In 2019, around 1.9 billion dollars were lost in fraudulent transactions [14].

Fraudulent transactions were traditionally detected by a static rules-based system built by fraud analysts. The rules were created based on data analysis and fraud analysts' expertise. Yet, with the increasing democratization of machine learning tools and the decreasing cost of processing and storing data, fraud detection started to move away from rule systems and closer to machine learning models. The advantage of these models is that they are much more accurate at detecting complex patterns and much more flexible to small changes in fraud activity. Today, machine learning is an essential tool to catch fraudsters and prevent fraud losses effectively [3].

One way of using machine learning to detect fraud is through supervised models. As the name suggests, these models learn fraudulent patterns from historical transaction data. To learn these patterns, supervised models require labels indicating whether a transaction was fraudulent or legitimate and these labels can arrive in two different ways. Firstly, if the fraud detection system in production flags a certain transaction as suspicious, human analysts will review it and mark it as either fraud or legitimate. Usually, these labels arrive to the system in a short period of time. Secondly, if a person suspects that he or she was a victim of fraud, they can contact the bank and initiate a chargeback process. Depending on the transaction amount, an investigation takes place, and in the event of transactions being verified as fraudulent, funds are returned to the cardholder. These chargeback labels can take days, weeks or even months to arrive.

Ultimately, most transactions will not go through either of these labeling processes and, in these cases, transactions are assumed to be legitimate. Since we cannot be sure that these transactions are indeed legitimate, the assumption of legitimacy is adding noise to the labels. Yet, it is not the only source of label noise in real world financial fraud data sets. Particularly, label noise can appear from the following sources:

- Fraud analysts can make mistakes when introducing labels manually in the system. In particular, when a fraud analyst contacts a client to verify a transaction's legitimacy, it is common to manually propagate the label to similar transactions, which can cause label errors.

- Analysts reviews are subjective, and thus two different analysts can make different decisions about the same transaction. This phenomenon is known as inter-expert variability. The judgement of whether a transaction is fraudulent can vary between analysts depending on their past experience. This subjectivity on classifying transactions can lead to inconsistencies in the labels.

- Communication problems with the client. Both analysts and clients can misunderstand the conversation and lead to the insertion of mislabeled transactions into the system.

- Transactions that are in white-lists: these cases are not alerted by the system and therefore, they will never be reviewed by a fraud analyst.

- Labels are usually not registered alongside the transaction data and thus, these data sources need to be joined. When the tables do not have a unique transaction identifier, which is not uncommon, the data needs to be joined with fuzzy matching of fields such as transaction amount and timestamp. This fuzzy matching inevitably leads to errors in the labels.

- Fraud prevention systems also suffer from fraud themselves, which is called friendly fraud. In order to collect the money from chargebacks, some clients report fake fraudulent activity, which in turn brings false labels into the data.

1.2 Company history

Feedzai is a Portuguese company founded in Coimbra in 2008 by Nuno Sebastião, Pedro Bizarro, and Paulo Marques. It started as a software start-up offering a real-time analytics engine and, in 2011, focused its efforts on fighting financial crime in e-commerce and banking.

Nowadays, Feedzai handles fraud detection for ten of the world's largest banks, merchants, and payment processors, and it is considered the market leader in fighting fraud by merging software engineering with state-of-the-art machine learning techniques. It offers a risk management solution for transaction monitoring, account opening and anti-money laundering.

Feedzai's processes and scores five billion dollars' worth in transactions every day and around 30 million transactions go through its system. Its risk management platform detects around 82% of the fraudulent cards for the top card issuer, has 66% money recall for the top acquirer, and 0.02% chargeback rate for a global merchant [41].

Feedzai has several offices around the globe, including Coimbra, Lisbon, Porto, Silicon Valley, New York City, Atlanta, Hong Kong, and London.

1.3 Problem and contribution

Data scientist working at Feedzai need to deliver good model performance at extremely small False Positive Rate (FPR). It is even more extreme for banking clients, which usually require FPR smaller than 2%. However, as we have seen before, the real-life data sets accessible to Data Scientists are expected to contain some noise. Depending on the level of noise, it can be challenging to achieve a model performance that fits these Key Performance Indicators (KPI). In this context, it is essential to explore the destructive effects of label noise in fraud detection.

In summary, the main goal of this project is to measure the impact of different types of label noise in the supervised models commonly used at Feedzai. Particularly, we aim to answer the following questions in the context of fraud detection:

- How robust are supervised tree-based models to different types of label noise?
- What is the impact of label noise in model evaluation?
- Which hyperparameters contribute the most for model performance?

To address these questions, we add different types of artificial label noise to three financial fraud datasets and explore the effects of the noise in model performance, model tuning, and model prediction distributions. To the best of our knowledge, this

is the first in-depth study of the impact of different types of label noise in fraud detection machine learning models.

Chapter 2

Literature review

2.1 Noise in classification problems

In classification problems, noise can be either present in the attribute values or the class labels. Noise in the class labels is commonly mentioned in the literature as label noise or class noise, and it consists of instances that have observed labels that do not accurately reflect the ground truth.

Training supervised models with noisy features can be a challenging task. Label noise is generally more harmful since each feature can have different feature importance during model training and consequently, the impact on the model may vary in many levels [55]. In contrast, when training a model, the algorithm relies heavily on the label. Therefore, it has a greater impact on the training process [55].

Although having a good set of labels is essential for training supervised models, getting accurate and reliable labels is time-consuming, expensive, and difficult, especially in Big Data problems. For this reason, there has been an increase of datasets being labelled by cheap, non-expert labelling on crowdsourcing services such as the Amazon Mechanical Turk [30, 44, 48, 54], resulting in extremely noisy datasets.

Certainly, crowdsourcing is not the only source of unreliable data, according to Frénay and Verleysen [16], label noise can come from several sources, such as:

- Data of poor quality;
- Insufficient information provided to the expert labelling the data;
- Errors made by the labeller;
- Subjectivity of the labelling task;

- Communication/encoding problems.

There are slightly different taxonomies for label noise in the literature, depending on the author. However, in their in-depth review of label noise, Frénay and Verleysen [16] suggest the following three categories:

1. Noise Completely at Random (NCAR): symmetric noise which occurs independently of the true class and the values of the instance features;

2. Noise at Random (NAR): asymmetric noise that only depends on the true label. This noise reflects the cases where some classes are more likely to be mislabeled than others;

3. Noise Not at Random (NNAR): occurs when the mislabelling probability depends on the feature values.

Experts still debate what type of noise is more prevalent in real datasets and which assumptions about this noise are more realistic [16]. For that reason, it is crucial to understand how different types of label noise impact model performance.

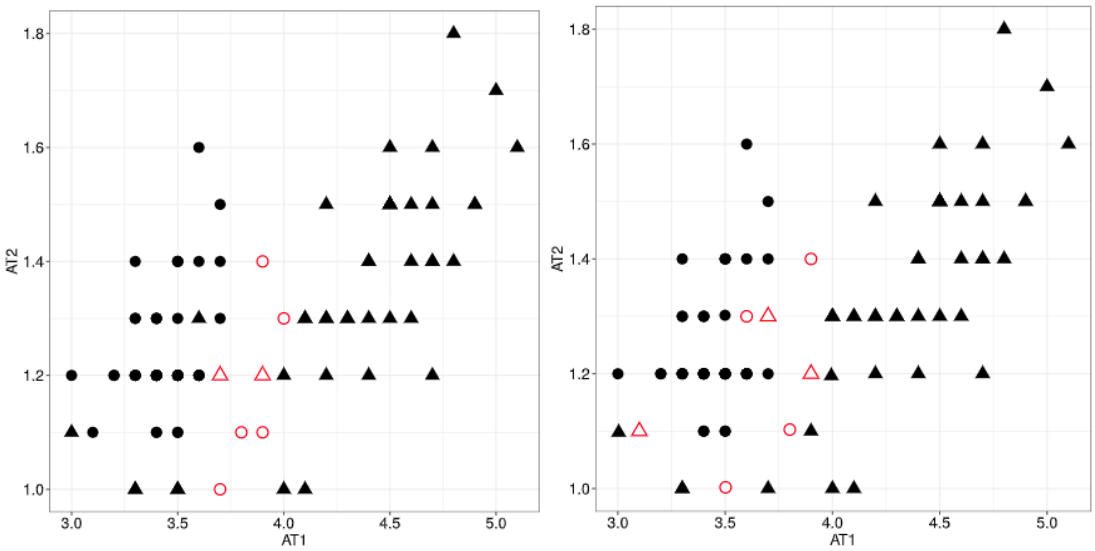
2.2 Artificial label noise injection methods

When studying label noise, it is extremely rare to find noisy datasets with the noise ground-truth. In these cases, the usual solution is to add artificial noise to real datasets. However, defining the assumptions that underlie the noise generation is not straightforward and different authors solve this issue in different ways. Using the taxonomy proposed by Frénay and Verleysen [16], we can divide the literature based on the noise category.

The injection of NCAR label noise is the most common method used in studies related to label noise [10, 20, 39, 40, 50]. This type of label noise is also the simplest to implement. To add noise, these authors sample instances uniformly and flip the label of these instances. Thus, the noise is added independently of the features and the label.

In studies simulating NAR noise scenarios, the label noise is injected in pairs of classes, where the label is flipped within the pair. The explanation behind this type of noise is to affect the class pairs that are more frequent classes since these classes are more likely to be misclassified [55, 56].

However, it is very likely that real noisy datasets do not only depend on the target label but also on the values of the attributes since examples harder to manually classify will be noisier than easier examples. The following three noise injection methods take this into consideration and target the examples closer to the classification boundary.



(a) NNAR NL **(b)** NNAR NN

Figure 2.1: Example of the labels that were disturbed in NNAR NL and NNAR NN injection methods, the region of points affected are located in the decision boundary. In the x axis, it is the attribute 1 (AT1) and in the y axis the attribute 2 (AT2). The red examples are the instances that were flipped by each injection method. The shape of the points correspond to the class. Figure taken from [19].

The Linearwise method proposed by Chhikara and McKeon [8] uses a Linear Discriminant Analysis (LDA) classifier to define the decision boundary.

The Non Linearwise injection method was first introduced by Garcia et al. [19] and uses a Support Vector Machine (SVM) with a Radial Basis Function to estimate the decision boundary. In this case, the non linearity of a dataset is quantified by the radial margin of each data point. The candidate instances to be flipped are the ones that have the lowest distance. These are the instances in the borderline and which a SVM model would be more uncertain about their prediction.

Finally, there is the Neighbourwise injection method, which was based on Tin Kam Ho and Basu [51] and first used by Garcia et al. [19]. For each instance, it calculates the ratio between the smallest intra-class distance and the smallest inter-class distance. The intra-class distance is the distance to the closest sample of the same class, whereas the inter-class is the distance to the closest sample of the opposite class. The distance metric chosen will depend on the dataset. If it only contains numerical features, the Euclidean distance might be used. In contrast, if the dataset contains both numerical and categorical features, the Gower distance [26] should be preferred. The candidate labels to be corrupted by this method are the ones with the highest ratios since these instances are more likely to correspond to borderline examples. The reasoning being that they are closer to examples with the opposite label (low inter-class distance) and further away from examples with the same label (high intra-class

distance), and therefore are probably close to the decision boundary.

2.3 Impacts of label noise

Even though we do not yet know the exact characteristics of the label noise present in real datasets, it is widely known that label noise can have severe impacts on machine learning tasks. The most prevalent consequence of label noise is the negative impact on classification performance. This has been extensively described in the literature and supported by both theoretical and empirical evidence. The impact level on model performance depends on the type and rates of label noise, the learning algorithm and the characteristics of the data [16].

The impact on model performance gets worse in imbalanced datasets. Folleco et al. [13] trained Decision Tree (DT) and RF models in noisy datasets and measured the impact of label noise in the Area Under the Curve (AUC). The authors concluded that both models were negatively impacted by label noise.

Other consequences are the increase in model complexity, leading to overfitting, and the need for more training data. Both issues result in higher computational complexity, larger models and a decrease in model interpretability [7].

Although direct model impact is the easiest consequence of noisy labels to understand, other tasks can have misleading results and lead to wrong conclusions. Examples include estimating observed class statistics, estimating class frequencies (e.g. disease prevalence), and doing feature selection [15]. For feature selection in particular, the techniques that use label information can generate unreliable and unstable results in the presence of label noise [15].

Despite all negative impacts caused by label noise, some authors empirically show an improvement of model performance when adding artificial label noise of the NCAR noise because it increases the variability of the training sets [4, 36–38, 52].

2.4 Machine Learning modeling

In order to solve complex classification problems, such as fraud detection, we need to find a way to model the patterns in the data that tell us whether a transaction is legitimate or fraudulent and, to do so, we can train a machine learning algorithm. Machine learning algorithms can either be supervised, where the algorithm is trained with labeled data, or unsupervised, where the algorithm is not provided with labels and it tries to find patterns in the data in order to group similar data points. We will focus on supervised algorithms, namely tree-based supervised algorithms, since these are extensively used at Feedzai.

2.4.1 Tree-based supervised models

Tree-based models are commonly used in classification and regression problems. These models capture complex non-linear relationships between features and are easily interpretable [27].

2.4.1.1 Decision tree

A decision tree is a type of supervised learning model, and the most popular algorithms are CART [6] and C4.5 [42]. Decision trees are composed of a root node, intermediate nodes, and leaves. At each node, a splitting criteria is used to find the feature and split-point that better separates the classes. This occurs at every node of the tree until the model reaches the maximum of discrimination. In order to avoid trees that grow indefinitely, we can specify the maximum depth or the minimum number of samples per leaf.

The most popular splitting criteria in decision trees are the Gini coefficient (Equation 2.1) and the Shannon Entropy (Equation 2.2). Both of these metrics quantify impurity at each node, where a node is considered pure if all elements belong to a single class. Furthermore, the best split will be the one that minimize these metrics, meaning that the child nodes will have the lowest impurity possible.

$$G(X) = \sum_{i=1}^{n} P(x_i)(1 - P(x_i)) \tag{2.1}$$

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i) \tag{2.2}$$

Information Gain (IG) is used to determine which feature gives the maximum information regarding a class using the Shannon Entropy. At each node in the decision tree, a new split is chosen by determining the feature and splitting point that yields the biggest decrease in impurity between the parent and its (weighted) children nodes.

A common problem with decision trees is that they are prone to overfitting. In order to reduce the likelihood of overfitting to the training data, pruning can be done in decision trees by removing nodes that do not provide additional information. This is very useful to reduce model complexity and consequently, improve model generalisation in unseen data. The amount of pruning done can be defined by using a measure called Cost-Complexity Pruning (CCP) and it is defined as:

$$R\alpha(T) = R(T) + \alpha|T| \tag{2.3}$$

where α is the complexity parameter, $R(T)$ is typically the misclassification rate, and $|T|$ is the number of terminal nodes in T. It uses the goodness of fit and tree size to find the subtree that minimises $R\alpha(T)$ [6].

2.4.1.2 Ensemble methods

The major disadvantage of using DT is its high variance when small changes occur in the data, resulting in very different models. For this reason, better accuracy and stability are often achieved when using a combination of simpler decision trees instead of a single one. This method is called an ensemble. In ensemble models, there are three ways of combining base learners, namely, bagging (also called Bootstrap Aggregation), boosting, and stacking methods. For the scope of this section, we will focus on bagging and boosting methods as they are the most used at Feedzai.

Bagging is a technique that reduces the variance of an estimated prediction function and works exceptionally well in conjunction with decision trees because of their high-variance and low-bias. In bagging, bootstrap samples are drawn by randomly sampling with replacement from the entire data. On each independent sample, typically of the same size as the original dataset, a decision tree is trained. The predictions of all these decision trees are then aggregated by majority voting or averaging. Typically, increasing the number of samples (and therefore decision trees) helps reduce variance of the ensemble. Therefore, it is often desirable to select a large number of trees (until performance eventually stops improving). An example of a bagging ensemble model is the RF [27].

In the case of RF, the data is sampled using the bootstrap technique and the number of features considered at each split point is also randomly selected. The three main principles of the random forest are to decrease the variance through the use of bootstrap samples and increase the variety between trees with the selection of random features at splitting and the sampling variety [27]. The final prediction will be the aggregated result of every tree in the forest. More specifically, in the RF Classifier it is the class which have the most votes across all trained trees in the forest (majority vote) while for RF Regressor case, the final prediction is generally the average given by all trained trees in the forest. The number of features and CCP are parameters that can be tuned [27].

For boosting models, in contrast to the bagging technique, the model training is sequential and additive. At every new iteration, the weak learners learns from the examples that were misclassified in the previous iteration. Adaboost [17] and LGBM [32] are examples of boosting models. The model initiates the training with each data point having the same weight. At each boosting step (or tree), the weights will be modified and the classification algorithm is reapplied to the weighted observations. The misclassified examples by the previous iteration will have a higher weight, while for

correctly classified examples the weight will be decreased. Subsequently, examples that are harder to classify will receive higher weights as iterations proceed [27]. The final prediction will be the weighted vote of the trees.

A more complex type of boosting algorithm is gradient boosting. The main idea is that in each round of boosting, a tree model is trained to regress the gradient of the loss of the model at the previous iteration. This allows us to train our ensemble model to minimize different loss functions. Adaboost can, in this case, be reinterpreted as a gradient boosting algorithm minimizing an exponential loss function.

LGBM is very similar to Extreme Gradient Boosting (XGBoost), but it stands out for its time and memory optimizations. The main features in LGBM that decreases the computational time used for training a model, and the memory consumption are the use of histogram-based buckets, Exclusive Feature Bundling (EFB), and Gradient-based One-side Sampling (GOSS) methods [32]. When calculating the best split point, the value of the features are sorted, and the continuous features are transformed into discrete features.

2.4.2 Robustness to label noise

Tree-based models are generally robust to NCAR if tuned correctly [10, 20, 39, 40]. Different regularisation techniques, such as post-pruning process or early-stopping criteria can avoid overfitting to instances that have noisy labels [16].

Different studies have compared robustness to label noise between bagging and boosting models. Experiments showed that accuracy of RF is less impacted by high levels of NCAR than Adaboost [39]. This behaviour can be explained by the optimization strategy chosen by each algorithm. In the case of RF, the split criteria that is more robust is the gini coefficient in comparison to entropy (theoretical results are showed in [20]). Then for the loss functions, the log-loss is more robust than 0-1 and exponential loss functions, this can be explained because the log-loss (or entropy) concentrates relatively less influence in the misclassified instances and more evenly spreads the influence across the entire data, while the exponential loss, used in Adaboost, penalizes misclassified examples with a exponential factor [27]. Different studies have identified Adaboost as very sensitive to NCAR [10, 25, 39].

A summary about robustness and the most robust parameters to label noise for tree-based models are included in table 2.1.

The effect of model performance and robustness in datasets with Noise Completely at Random (NCAR) has been extensively studied with theoretical and empirical results and comparing robustness between bagging and boosting algorithms [10, 20, 39]. Few studies cover label noise at random (NAR) [20] however, to the best of our knowledge, the study of the effect of noise not at random (NNAR) on model

robustness is scarce in the literature. For that reason, a study that compares the robustness in a wide variety of artificial label noise scenarios (including random and not at random label noises) is necessary to quantify and compare between models trained on different label noises.

Table 2.1: Impact of label noise and parameter robustness in tree-based models.

Model	**Impact**	**Ref.**
DT	DT tends to learn from noisy labels, leading to models that are more complex and overfitting the noisy examples. The post-pruning process can increase model robustness by reducing the size of the trees. Regarding the split criterion, Gini coefficient is robust to NCAR whilst Shannon entropy is fairly robust to label noise. The leaf sample size should be increased if the noise rate is high. If a model is tuned correctly, DT can be robust to symmetric or asymmetric noise even without knowledge about noise rates.	[20, 40, 55]
Adaboost	Sensitive to label noise, as it gives higher weights to mislabelled instances comparatively with correctly labelled instances during model training. This way, the model tends to overfit the mislabelled instances. Limiting the number of iterations with early stopping criteria can avoid overfitting. In terms of the weight updating coefficient, using the Breiman's weights improves accuracy comparatively with Freund's weights.	[10, 25, 39]
XGBoost	Between different boosting models (Adaboost and Gradient Boosting Machine (GBM)), XGBoost was the most robust model in different levels of label noise (noise up to 20%) in binary classification.	[25]
RF	RF is robust to symmetric label noise under the large sample size limit at the node, but is sensitive to asymmetric noise. The minimum sample size per leaf nodes should be increased if there is high rates of class noise. Similarly to what happens to DT, the most robust split criteria is Gini index measure. The method for assign a class label should be majority voting criterion because this method is more robust to symmetric noise. In low levels of noise, RF can even benefit from it because it increases the variety between the trees in the forest. Adding the property to randomise split points can decrease the effectiveness of RF when label noise is present.	[10, 20, 39]

2.4.3 Evaluation

In imbalanced datasets, traditional metrics can fail to estimate model performance and capture relevant information. An example of such metric is accuracy. If only 1% of the observations are positive and the model predicts all observations to be negative, it would achieve an accuracy of 99%. This is not the best way of evaluating the model because it would perform poorly in classifying the positive class examples. This is commonly known as the accuracy paradox.

Using alternative metrics that consider the number of true positive class is better for evaluating the model performance in the class of interest. Examples of these metrics are described next, but first it is important to understand some core concepts.

The True Positive (TP) refers to a data point that was classified by the model as belonging to the positive class and is indeed of the positive class. The False Positive (FP) refers to a data point that was classified by the model as belonging to the positive class but is in fact of the negative class. A True Negative (TN) is an instance that the model classified as belonging to the negative class and is in fact of the negative class. Finally, a False Negative (FN) is a data point that was classified by the model as belonging to the negative class but is in fact from the positive class.

The recall (also called sensitivity, detection rate or True Positive Rate (TPR)) provides the rate of TP identified by the model out of all the data points belonging to the positive class. It is defines as:

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

The precision encapsulates the rate of TP identified by the model out of all the data points that the model classified as being of the positive class. Its formula is as follows:

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

The FPR represents the rate of FP given by the model out of all the data points belonging to the negative class. Its formula is:

$$FPR = \frac{FP}{FP + TN} \tag{2.6}$$

There are not many studies regarding which evaluation metrics are more effective when dealing with noisy datasets. Lam and Stork [34], verifies that when dealing with smaller percentages of random label noise the error in the Precision-Recall curve can still be correctly estimated as long as the classifier and the labeller make independent mistakes. According to Sheng et al. [47], empirical results show that AUC is more robust when dealing with larger label noise percentages, however, for smaller noise

percentages, accuracy seems to be a more suitable metric, but, as we explained before, accuracy is not the most appropriated metric for our use case.

The Receiver Operating Characteristic (ROC) curve is a graphical plot that accesses the ability of a binary classification model to classify the data points, and it is given by the plotting of recall against FPR for all threshold values. One of the most commonly used summary indices derived from the ROC curve is the AUC. The closer the curve is to the top left corner of the chart, the better the model is. A ROC curve that makes a 90º corner on the top left (100% AUC) indicates that the model can perfectly distinguish between the two classes. On the other hand, a ROC curve that resembles identity curve (50% AUC) has no discrimination capability between the classes. Finally, if the model makes a 90º corner on the bottom right (0% AUC), the model can perfectly distinguish between the classes, but is reciprocating them. For highly imbalanced datasets, such as the fraud detection use case, it is more relevant to focus on recall at low levels of FPR. To do so, the range of AUC can be summarised and standardised to a portion of the ROC on a range of interest, in our case, between 0% and 2% of FPR.

2.4.4 Hyperparameter search and importance

Hyperparameters tuning is a fundamental step to achieve well performing models. There are different techniques for optimising hyperparameters, such as grid search, random search, and Bayesian optimisation [53]. In grid search, all possible combinations for the set of hyperparameters are tested, making it very expensive in terms of time and space complexity. For random search, a number of trials is set, and for each trial a random combination of values is chosen for the hyperparameters. Intuitively, it is very likely that given a sufficient number of trials, random search can find models with good performance, making it very cheap and efficient. On the other hand, Bayesian search generally involves using a Gaussian Process to fit the classifier's outcome and measure the correlation between tasks which can be exploited for the hyperparameter optimisation on a new task. This makes it so that Bayesian search can reach an optimal model faster than the other two previously discussed methods. However, Bayesian search has a risk of reaching local optima on smaller datasets [33] and is highly dependent of the parameters chosen for the initialization [12].

Since the datasets used for our work are relatively small compared to the usual size of the datasets used at Feedzai, and we did not want to limit the search space based at the start of the hyperparameter search, and in order to have a more fair comparison between different datasets and noise percentages, we chose to use random search over a Bayesian approach for our work.

Recent methods quantify the parameter importance by taking advantages of RF models, by fitting a RF regressor model over the set of hyperparameter' values that

were used to trained a classifier and using the performance values it outputs as the target value (e.g. accuracy or AUC). After training the regressor model, the hyperparameter importance can be estimated by using Mean Decrease Impurity (MDI) or Functional ANOVA (fANOVA).

As explained in its original paper [5], MDI calculates the importance of each feature as the sum of the number of splits across all trees in the model that include the feature, weighted by the number of instances it splits.

In fANOVA framework [28], the priors distribution for the hyperparameters are inferred and it determines how individual parameters and interaction between set of parameters contributes to the variance of the performance. A parameter that is responsible for a large fraction of the variance will be considered as important since it is necessary to set this parameter correctly in order to obtain good performance.

Chapter 3

Methodology

In this project, we studied the effect of artificial label noise on model performance and tuning in three different publicly available fraud detection datasets. Because we were interested in the fraud detection use-case, we focused the analysis on the most used models at Feedzai, namely, Random Forest (RF) and LightGBM (LGBM). We also investigated how robust the hyperparameters of these models are and how the label noise impacts model evaluation.

In this chapter, we will start by describing the datasets used in this analysis and how they were processed. We will follow by presenting the different types of artificial label noise, and the techniques to inject such noise into our datasets. Finally, we will focus on the models used, the number of models trained, the work done for tuning the parameters of these models, and how we assessed their performance.

3.1 Fraud detection datasets

The three datasets used for this experiment, IEEE-CIS Fraud Detection (IEEE) [29], Synthetic Data from a Financial Payment System (Banksim) [35, 49], and Credit Card Fraud Detection (CC) [9] are available on Kaggle [31], a popular platform where data scientists and ML practitioners participate in Data Science competitions.

The IEEE dataset comes from the Vesta Corporation, a company that provides e-commerce payment services. It contains real-life transactions and a wide variety of anonymizated features [29]. Banksim is an agent-based simulator of bank payments, and it was generated from a sample of aggregated transactional data provided by a Spanish bank [35, 49]. Finally, the CC dataset contains credit card transactions in Europe after a Principal Component Analysis (PCA) transformation for confidentiality purposes [9].

The datasets used in our experiment have a small percentage of fraudulent events (between 0.1% and 4.0% of the transactions), and the binary target variable identifies whether a transaction is fraudulent or not. The characteristics of the datasets are included in the Table 3.1.

Table 3.1: Datasets considered for the artificial label noise experiment.

Dataset	# Examples	# Attributes	Fraud rate	Time span
Banksim	594,643	9	1.211%	6 months
CC	283,726	30	0.167%	2 days
IEEE	590,540	432	3.499%	6 months

3.1.1 Data preprocessing

We performed several preprocessing steps before generating the artificial label noise and model training. For each dataset, we checked the number of duplicate lines, missing values, distinct values, and categorical levels. Then, based on the results of this data exploration, we performed some specific transformations to each individual dataset.

For the Banksim dataset, we removed three columns because they were either duplicated or only had one value.

In the CC dataset, we verified that there were duplicated examples, and we excluded them from the data. We transformed the categorical features using one-hot encoding before generating the label noise and training the RF.

Finally, for the IEEE dataset, missing values were imputed with 'others' for categorical features and '-999' for numerical features. The categoricals were encoded with ordinal frequency encoding in order to avoid the increasing number of attributes in the data and consequently, increase the computational time when training the models. Due to the high number of features in IEEE dataset, we performed feature selection on categorical features by removing the columns that had high cardinality or a constant value.

3.1.2 Data split

Fraud patterns are highly difficult to model, not only are they highly complex, but fraudsters keep adapting and finding new ways to beat the models, making it an adversarial problem. We also need to take into account the concept drift that occurs, as transaction patterns shift over time. It usually happens that a model that was very good at distinguishing fraudulent transactions from the legitimate ones once, can rapidly see its performance degrade and be in need of retraining. Therefore, using data from the future can leak information while training a model and it is likely to give an optimistic model error estimation.

For this reason, using forward-validation schemes that keep the temporal order of observations is necessary to ensure that the time dependencies in the data are respected, are inherently free of look-ahead bias, and that the estimation of the model performance is more realistic. There are several types of data splitting techniques that follows a forward-validation scheme in the literature (for more details see [2]).

Table 3.2: Time span and number of transactions for train, validation and test set for each dataset.

Dataset	Set	Time span (days)	Number of transactions	Fraud (%)
Banksim	Train	131	416250	1.263%
	Validation	16	59464	1.115%
	Test	32	118929	1.076%
CC	Train	1.5	198608	0.184%
	Validation	0.125	28372	0.116%
	Test	0.292	56746	0.130%
IEEE	Train	119	413378	3.517%
	Validation	20	59054	3.490%
	Test	41	118108	3.441%

Taking this into account, we split the data into train, validation, and test sets sequentially, respecting the temporal order of observations. The chosen ratio for the split was 70% for the train set, 10% for the validation set and 20% for the test set. The time span and number of transactions for each set is included in Table 3.2.

3.2 Types of artificial label noise

In this project we generated five different noise scenarios:

1. Noise Completely at Random (NCAR) - labels are flipped at random, independently of the class.

2. Noise at Random in the negative class (NAR0) - the legitimate labels are flipped at random. Note that in fraud detection, this is the majority class.

3. Noise at Random in the positive class (NAR1) - the fraudulent labels (i.e. the minority class) are flipped at random.

4. Noise Not at Random Neighbourwise (NNAR NN) - labels are flipped based on the distance to their neighbours' distance and their labels [19].

5. Noise Not at Random Non Linearwise (NNAR NL) - labels are flipped based on the radial margin of a SVM model [19].

These noises are examples of the three types of label noise described in the literature [16]. NCAR is a noise completely at random, Noise at Random in the negative

class (NAR0) and Noise at Random in the positive class (NAR1) are noises at random, and finally, Noise Not at Random Neighbourwise (NNAR NN) and Noise Not at Random Non Linearwise (NNAR NL) are noises not at random.

3.2.1 Injection of artificial label noise

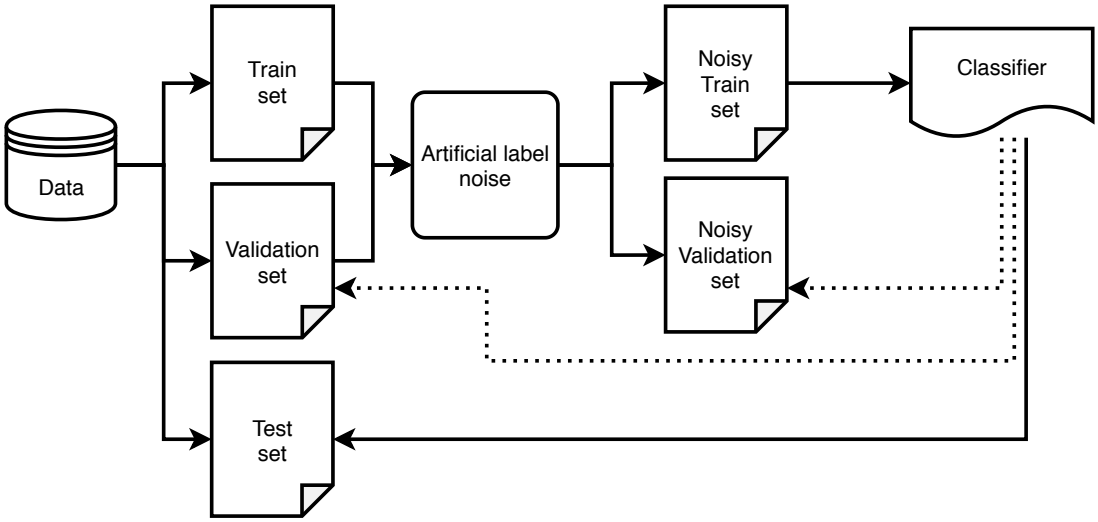


Figure 3.1: Experiment setup.

We have disturbed 5%, 10%, 20%, 30% or 40% of train and validation sets' labels, while the test set remained unchanged for evaluation purposes (Figure 3.1).

The number of noisy labels depended on the noise scenario, for NCAR and NNAR, we affected 5%, 10%, 20%, 30% or 40% of the total number of labels, whilst for NAR, we affected the same percentages out of the total number of each class. The number of changes discriminated per class and the new fraud rate for the training set is included in Table A.2.

The noises NNAR NN and NNAR NL were applied to training and validation set together to guarantee that noise is applied consistently in the two sets.

For the NNAR NN, given that we have both numerical and categorical features, we used the Gower distance [26] to calculate the pairwise distance for every instance. This was the same distance used by the original authors of this noise scenario [19]. Then, we calculated the ratio of intra-class/inter-class for each instance of the dataset. The intra-class distance is the distance to the closest sample of the same label, while the inter-class is the distance to the closest sample of the opposite class. Examples that have high ratios are likely to be closer to the decision boundary since the points are located closer to the examples of opposite class than to points from its own class.

Afterwards, the top % of instances with higher ratios were flipped according to the desired noise level.

For the NNAR NL, we used a SVM model with a radial kernel with the balanced class weighted to overcome the issue of imbalanced classes. The class weight parameter penalises an example differently whether it belongs to the minority or majority class. We set the class weight as 'balanced', so the minority class weight is inversely proportional to the class frequency. The remaining parameter used the default values. Similarly to NNAR NN, this noise was applied over the training and validation set together. Therefore, the model was trained over the union of those sets and we measured the precision and recall of the SVM model on that data. The results are presented in Table A.1.

Each configuration was tested three times for three different seed values. In total, we generated 330 noisy training and validation sets.

3.3 Modeling

3.3.1 Model training and hyperparameter tuning

We trained a RF and LGBM for the baseline and the different levels of label noise, and tuned the hyperparameters using random search. We tested 100 different sets of model parameters. In total, we trained 3,300 different models.

In order to check if the optimal model hyperparameters change when label noise is added, for each parameter combination, we trained the models with noisy data but evaluated them in both the original and noisy validation sets (Figure 3.1). This way, we were able to check whether the evaluation in the noisy validation was a good proxy for the original validation performance and whether the parameters found in the noisy validation set were robust to label noise.

The parameters selected for hyperparameter tuning were based on their ability to minimize model overfitting and complexity (i.e. CCP for RF or Lambda L1/L2 for LGBM), as these are the main effects linked to label noise.

In the case of the LGBM implementation, it is possible to choose between three types of boosting models: the traditional gradient boosting, DART [43], and GOSS [32]. The features are discretized into buckets and the size of the bin is a parameter of the model that can be tuned and it will affect the computational time, as a higher number of bins will be more computationally demanding. EFB which consists on a bundle exclusive features in a sparse feature space, leads to a reduced number of features in the training set. Finally, GOSS increases computational efficiency without losing the accuracy of the information gain estimation. The instances with different gradients will have different sampling ratios, meaning instances with larger gradients are kept and will contribute more to the information gain, while instances with smaller gradients will be randomly dropped. The approximation works well because the data is subsampled from points that have lower gradients and would contribute less to the

log-loss function calculation [32].

The DART method was based on the idea of dropout used in the context of Deep Learning, which consists of muting complete trees to reduce bias [32].

When training a model with the LGBM implementation of gradient boosting model or a DART, it is possible to add a bootstrap method in the boosting models and benefit from properties of subsampling the training set observations. Both the bagging fraction and the bagging frequency can be tuned.

A summary of the parameters and their values used in the experiment are described in Table 3.3.

We set the number of iterations (or trees) for RF to be 200 for Banksim and CC and 100 for IEEE. IEEE had fewer trees due to the fact of having more data and thus being more computationally expensive than the other two datasets.

3.3.2 Model evaluation

The performance of the models was measured using partial AUC (pAUC) with an FPR range between 0% and 2%. This evaluation metric is meaningful for transaction monitoring use-case for banking solutions, as we explained in section 2.4.3.

An adaptation of the Equalized Loss Accuracy (ELA) was considered as it takes into account both performance and robustness of models. It is relevant to analyse both of these concepts together to study the behaviour of classifiers in the presence of noise [46]. We adapted the metric Equalized Loss Accuracy (ELA) [46] to consider the pAUC score instead of the accuracy score:

$$ELpAUC = \frac{(1 - pAUC_{x\%})}{pAUC_{0\%}} \tag{3.1}$$

where $pAUC_{x\%}$ denotes pAUC at a noise level of x%. A low value means that the model has a high performance in terms of pAUC in the baseline and this value does not drop much between label noise levels.

We also inspected the distribution of the model predictions in the test set. For that, we performed a probability adjustment (Equation 3.2) [11]. This step was necessary to adjust the predicted probability due to the class weight parameter used while training the RF or LGBM.

$$Adjusted\_prob = \frac{Prob}{Prob + class\_weight * (1 - Prob)} \tag{3.2}$$

Table 3.3: Hyperparameter values and description used in RF and LGBM models.

(a) RF Hyperparameters

Hyperparameter	Values	Sample Distribution	Description
Split criterion	{entropy,gini}	Uniform	Split quality function.
Max. depth	[5,50]	Uniform	Maximum depth of the tree.
Min. samples split	[5, 100]	Uniform	Minimum number of examples required to split the internal node.
Min. samples leaf	[5,200]	Uniform	Minimum number of examples required to create a leaf.
Feature fraction	[0,1]	Uniform	Fraction of features sampled per node.
Max. samples leaf nodes	[5,200]	Uniform	Maximum number of examples per leaf.
CCP alpha	[1e-8,1]	Log uniform	Complexity parameter (α) used for CCP.
Max. samples	[0.4,1]	Uniform	Maximum number of examples per tree.
Class weight	[1,100]	Uniform	Weight given to the positive class.

(b) LGBM Hyperparameters

Hyperparameter	Values	Sample Distribution	Description
Number of iterations	[1,500]	Uniform	Number of boosting iterations.
Lambda L1	[1e-8,10]	Log uniform	L1 regularisation.
Lambda L2	[1e-8,10]	Log uniform	L2 regularisation.
Number of leaves	[5,500]	Uniform	Maximum number of leaves per tree.
Learning rate	[1e-3,1]	Log uniform	Shrinkage rate.
Min. samples in leaf	[5,200]	Uniform	Minimum number of examples per leaf.
Max. bin	[5,500]	Uniform	Max. number of bins when discretizing the feature values.
Fraction of features	[0.4,1]	Uniform	Fraction of features sampled per iteration.
Boosting type	{gbdt,goss, dart}	Uniform	Traditional gbdt or apply LGBM optimisations.
Min. sum hessian leaf	[1e-3,200]	Uniform	Minimum sum hessian per leaf.
Max. depth	[5,50]	Uniform	Maximum depth of the tree.
Bagging fraction	[0.4,1]	Uniform	Fraction of examples sampled per node.
Bagging frequency	[1,7]	Uniform	Interval of iterations which occurs bagging. If 1, it will occur at every interaction.
Class weight	[1,100]	Uniform	Weight given to the positive class.

3.4 Programming language and packages

The whole experiment was performed in Python. The artificial label noise injection methods were implemented by us, apart from the Gower distance used in NNAR NN [24]. For the modelling part, we used the Microsoft LightGBM package [21, 32] and for RF the scikit-learn package implementation. We used Optuna [1, 23] to tune every model with random search and to calculate hyperparameter importance.

For result visualisations, we used Seaborn package [22].

Chapter 4

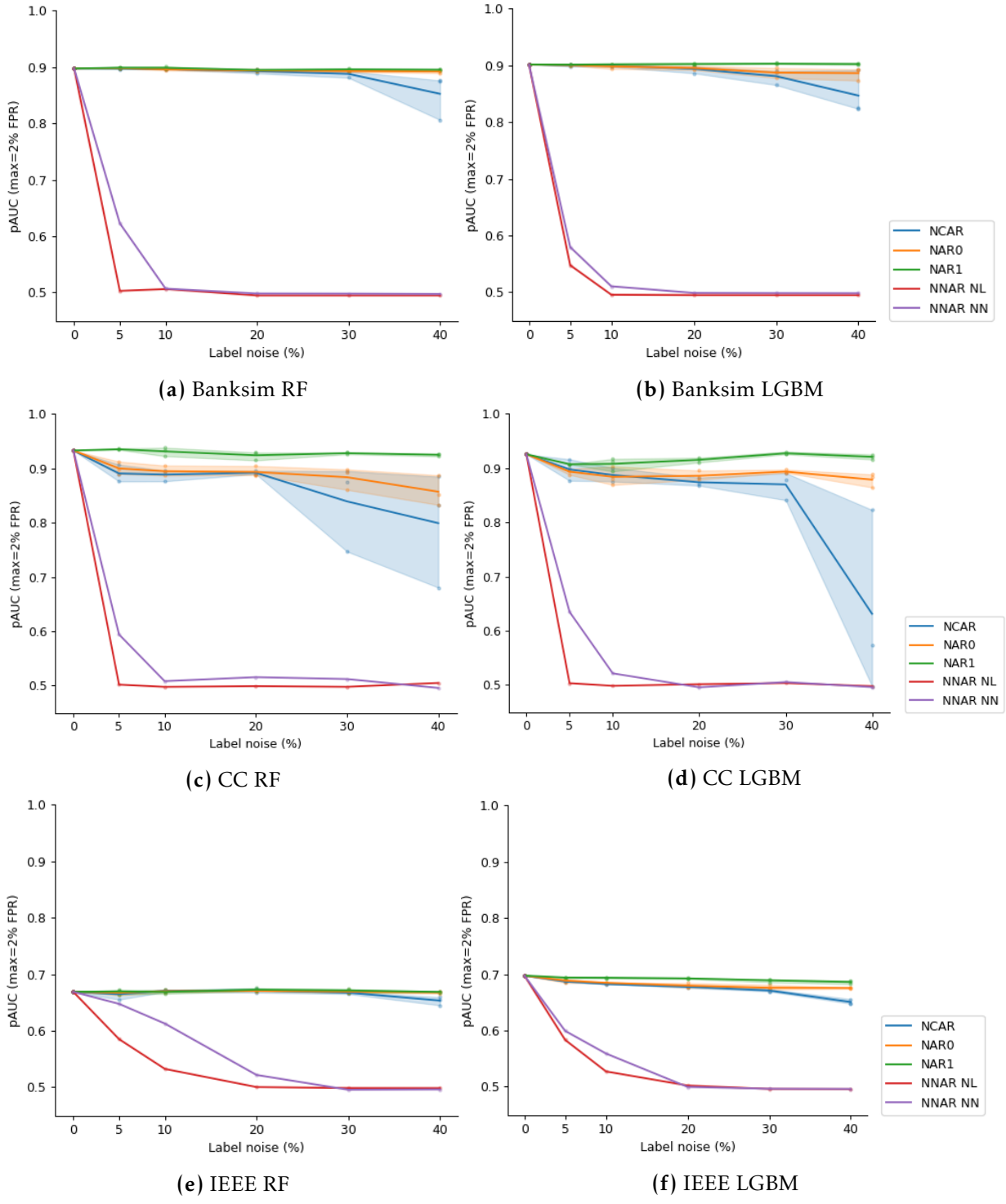
Results and discussion

4.1 Model robustness

We analysed the impact on pAUC for the best models selected in the noisy validation set and observed that the performance of models trained in the presence of random noise, such as NCAR, NAR0 and NAR1, was comparable to the performance of the baseline at 0% noise (Figure 4.1). In fact, the pAUC did not decay for percentages up to 20% of label noise. For higher percentages of NCAR and NAR0, the model performance varied more between the different random seeds, and for NCAR the average of pAUC decreased (Figure 4.1).

From all the label noises injected in our datasets, the NAR1 which is the noise affecting the minority class, was the least destructive. In this case, the pAUC remained unchanged across all percentages and datasets (Figure 4.1). We believe that the models remain robust at all levels of the NAR1 noise because it flips a much lower number of labels than all the other methods. Recall that, at each level, this noise flips a percentage of the fraudulent labels, which is the minority class. Thus, in absolute terms, it introduces much less noise than the other methods. In other words, it is necessary to disturb a much higher number of fraud labels in order to impact model training.

In contrast, both models were sensitive to the noise not at random. In these two noise types, both RF and LGBM suffered a significant drop in performance, as can be observed in Figure 4.1). In these cases, the performance reached a pAUC score close to random at low percentages of noise, meaning that the models are no longer capable of distinguishing between the two classes. Looking into the performance decay at different levels of label noise for NNAR NL and NNAR NN, we could observed that NNAR NL was the most destructive type of label noise, indicated by a steeper drop of performance that occurred in the three datasets.



(a) Banksim RF

(b) Banksim LGBM

(c) CC RF

(d) CC LGBM

(e) IEEE RF

(f) IEEE LGBM

Figure 4.1: Model performance in the test set per each noise type. The lines represent the mean of the three random seeds for NCAR, NAR0 and NAR1 cases. The filled areas show the minimum and maximum performance for the three random seeds and the data points represent the performance for each random seed.

Table 4.1: Best model performance and robustness in the unchanged test set for RF and LGBM when the models were tuned and selected in the noisy validation set.

(a) Banksim data

Model	Label noise	0%	5%		10%		20%		30%		40%	
		pAUC	pAUC	ElpAUC	pAUC	ElpAUC	pAUC	ElpAUC	pAUC	ElpAUC	pAUC	ElpAUC
RF	NCAR*	0.898	0.897	0.115	0.896	0.116	0.893	0.119	0.888	0.125	0.853	0.164
	NAR0*		0.898	0.114	0.896	0.116	0.894	0.118	0.893	0.119	0.892	0.120
	NAR1*		0.899	0.113	0.899	0.113	0.895	0.117	0.896	0.116	0.895	0.117
	NNAR NL		0.503	0.554	0.506	0.550	0.495	0.563	0.495	0.563	0.495	0.563
	NNAR NN		0.623	0.420	0.507	0.549	0.498	0.559	0.498	0.559	0.497	0.560
LGBM	NCAR*	0.902	0.900	0.111	0.899	0.112	0.894	0.118	0.881	0.132	0.847	0.170
	NAR0*		0.901	0.110	0.898	0.113	0.895	0.116	0.887	0.125	0.886	0.126
	NAR1*		0.901	0.109	0.902	0.109	0.903	0.108	0.903	0.108	0.902	0.108
	NNAR NL		0.548	0.502	0.496	0.559	0.495	0.560	0.495	0.560	0.495	0.560
	NNAR NN		0.580	0.466	0.510	0.543	0.499	0.556	0.498	0.556	0.498	0.557

(b) CC data

Model	Label noise	0%	5%		10%		20%		30%		40%	
		pAUC	pAUC	ElpAUC	pAUC	ElpAUC	pAUC	ElpAUC	pAUC	ElpAUC	pAUC	ElpAUC
RF	NCAR*	0.933	0.897	0.111	0.884	0.124	0.894	0.114	0.890	0.118	0.834	0.178
	NAR0*		0.900	0.107	0.895	0.113	0.893	0.114	0.884	0.125	0.857	0.153
	NAR1*		0.936	0.069	0.934	0.070	0.929	0.076	0.929	0.076	0.926	0.079
	NNAR NL		0.502	0.534	0.498	0.538	0.499	0.537	0.498	0.538	0.505	0.531
	NNAR NN		0.594	0.435	0.508	0.527	0.516	0.519	0.512	0.523	0.496	0.540
LGBM	NCAR*	0.926	0.898	0.111	0.887	0.122	0.874	0.136	0.870	0.141	0.631	0.398
	NAR0*		0.893	0.115	0.884	0.125	0.886	0.123	0.894	0.115	0.879	0.131
	NAR1*		0.907	0.100	0.908	0.100	0.915	0.092	0.927	0.078	0.921	0.086
	NNAR NL		0.503	0.536	0.499	0.542	0.502	0.539	0.504	0.536	0.498	0.542
	NNAR NN		0.635	0.395	0.522	0.517	0.496	0.545	0.505	0.534	0.496	0.544

(c) IEEE data

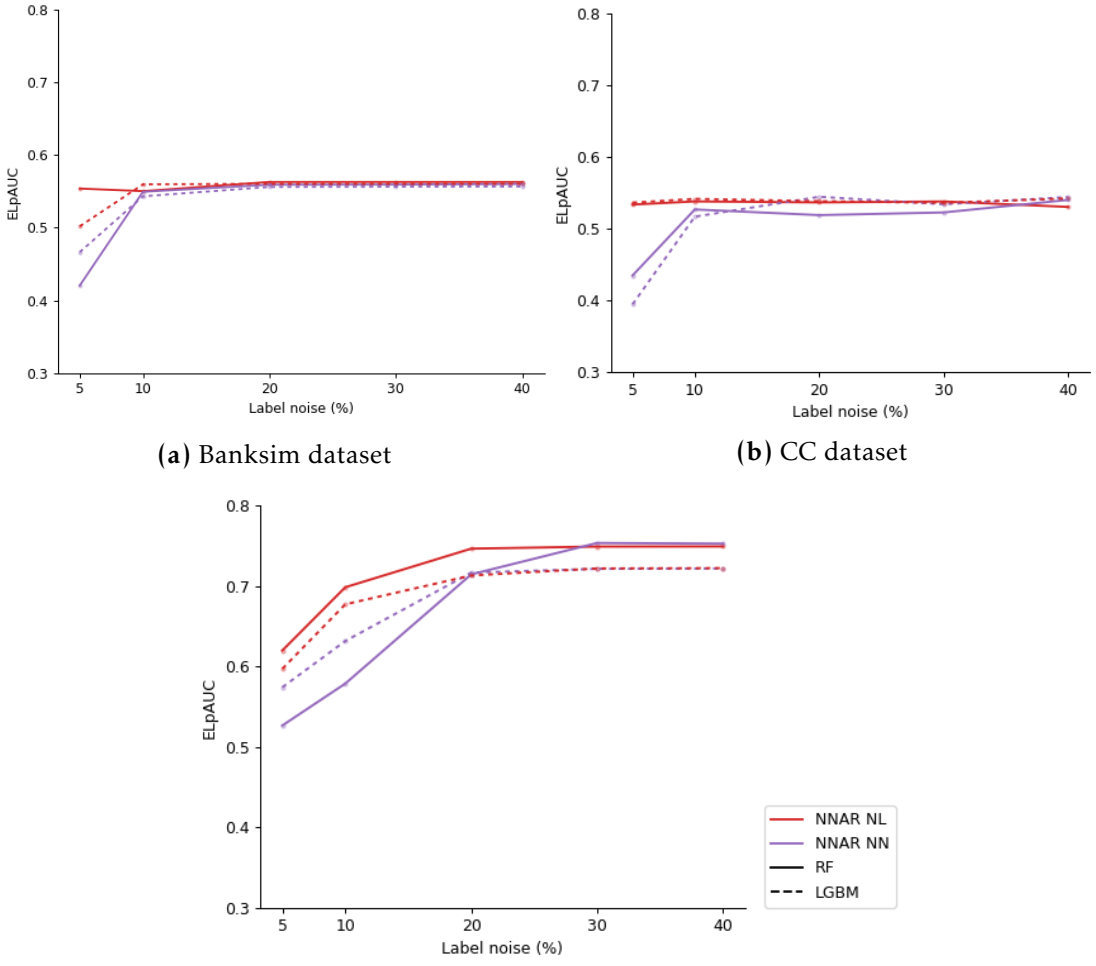
Model	Label noise	0%	5%		10%		20%		30%		40%	
		pAUC	pAUC	ElpAUC	pAUC	ElpAUC	pAUC	ElpAUC	pAUC	ElpAUC	pAUC	ElpAUC
RF	NCAR*	0.669	0.665	0.501	0.671	0.492	0.672	0.491	0.667	0.497	0.653	0.518
	NAR0*		0.667	0.498	0.670	0.493	0.671	0.492	0.669	0.494	0.668	0.497
	NAR1*		0.670	0.494	0.669	0.495	0.673	0.489	0.671	0.491	0.669	0.495
	NNAR NL		0.582	0.620	0.533	0.699	0.500	0.747	0.499	0.749	0.499	0.750
	NNAR NN		0.648	0.527	0.613	0.579	0.522	0.715	0.496	0.754	0.496	0.753
LGBM	NCAR*	0.698	0.687	0.448	0.683	0.455	0.678	0.462	0.671	0.471	0.651	0.501
	NAR0*		0.689	0.446	0.685	0.452	0.680	0.459	0.676	0.464	0.676	0.465
	NAR1*		0.694	0.438	0.694	0.439	0.693	0.441	0.689	0.445	0.686	0.450
	NNAR NL		0.583	0.597	0.527	0.678	0.502	0.713	0.496	0.722	0.496	0.723
	NNAR NN		0.599	0.574	0.559	0.632	0.500	0.717	0.496	0.722	0.496	0.722

[1]These values are the average of three random seeds.

Surprisingly, the NNAR NL noise drastically impacted model performance in all three datasets, besides the relatively poor performance of SVM model that generated this type of artificial label noise (Table A.1). Thus, it is not necessary to have a very discriminating decision boundary to efficiently target labels that negatively affect model training.

When we compared the performance decay for NNAR NL and NNAR NN between the three datasets, we observed that for CC and Banksim datasets, LGBM performance was lower at smaller percentages of noise than for the IEEE dataset (Figure 4.1). For the latter, the decrease of pAUC occurred more gradually. Interestingly, this dataset is the most realistic since it is the one dataset that covers a large time-window of real-life transactions, and have a good set of features (Table 3.1).

The results of pAUC and Equalized Loss pAUC (ELpAUC) for all datasets are described in the Table 4.1.

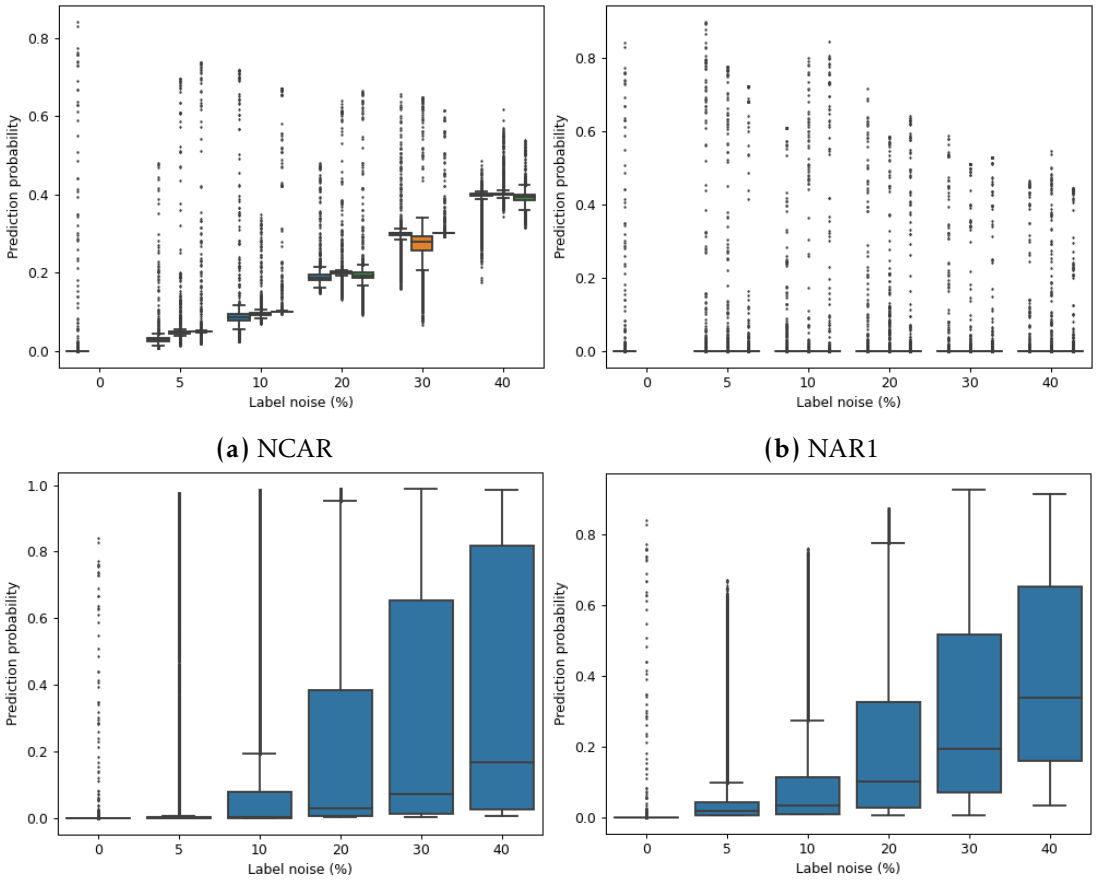


(a) Banksim dataset

(b) CC dataset

(c) IEEE dataset

Figure 4.2: ELpAUC measure for RF and LGBM models in presence of not at random label noise. Lower values of ELpAUC correspond to models that have better performance and robustness.



(a) NCAR **(b)** NAR1

(c) NNAR NL **(d)** NNAR NN

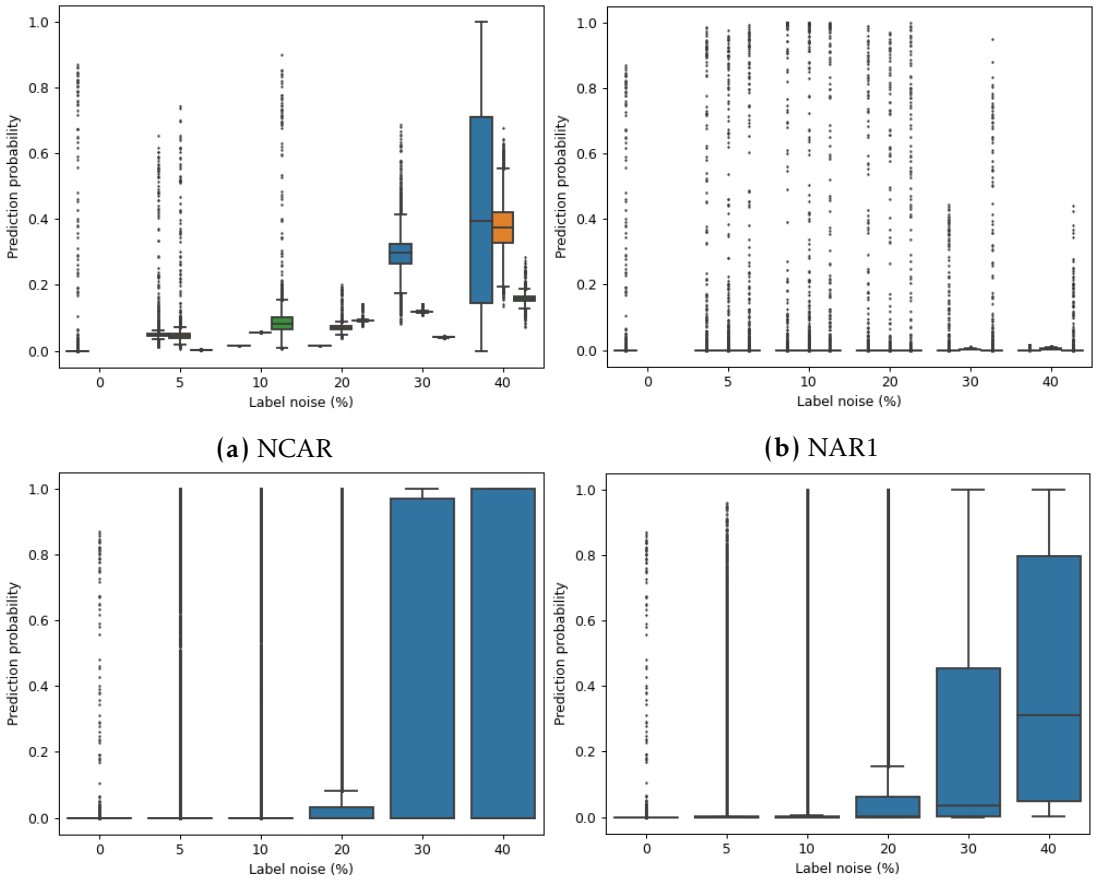
Figure 4.3: RF probability distribution for the selected best model in the noisy validation when in presence of NCAR, NAR1, NNAR NL and NNAR NN in CC test set. In the case of random label noises, each boxplot represents one random seed.

Intuitively, one would think that LGBM would be less robust to label noise than RF since boosting algorithms are more prone to overfitting the noisy examples in the training data. However, comparing the results between RF and LGBM, it is not evident which algorithm is the most robust to NNAR. Looking at Figure 4.2, the pAUC of each algorithm behaves differently depending on the dataset and level of label noise.

In the CC dataset case, the best performing algorithm for small levels of NNAR NN (up to 10%) was LGBM while for the Banksim and IEEE dataset was actually the RF.

For NNAR NL, in the CC dataset, there was no difference between RF and LGBM, whereas for Banksim, LGBM performed better but only for 5%. Finally for IEEE, LGBM was the best performing to all levels of noise.

In order to understand if label noise impacted model confidence, we inspected the model predictions for RF and LGBM for each type of noise. We verified that the probability distributions for NAR0 were consistently similar to NCAR and, thus, we



(a) NCAR **(b)** NAR1

(c) NNAR NL **(d)** NNAR NN

Figure 4.4: LGBM probability distribution for the selected best model in the noisy validation when in presence of NCAR, NAR1, NNAR NL and NNAR NN in CC test set. In the case of random label noises, each boxplot represents one random seed.

are only presenting the plots for NCAR case in Figure 4.3 and 4.4.

The level of confidence in the RF predictions decreases as we increase the number of noisy labels. The predicted probabilities for the noisy validation approach values close to 0.5 as we increased the level of NCAR, NAR0 and NNAR, meaning that the model increases uncertainty (Figure 4.3). For NAR1, as we are only affecting the positive labels, the model becomes more uncertain at predicting fraudulent cases (Figure 4.3b).

For LGBM, in NCAR and NAR0 scenarios, the probabilities tend to be skewed to the right when comparing to the baseline, and the level of skewness increases with the percentage of label noise. However, the distribution is distinct between seeds and the skewness in the probabilities is not verified for all cases (Figure 4.4). For NAR1, we observed a similar behaviour to the one described for RF for high levels of label noise, where the number of transactions predicted with higher probabilities decrease, and so, the distribution of probabilities is skewed towards zero (Figure 4.4b)

In general, we observe that for both algorithms and all types of label noise, the

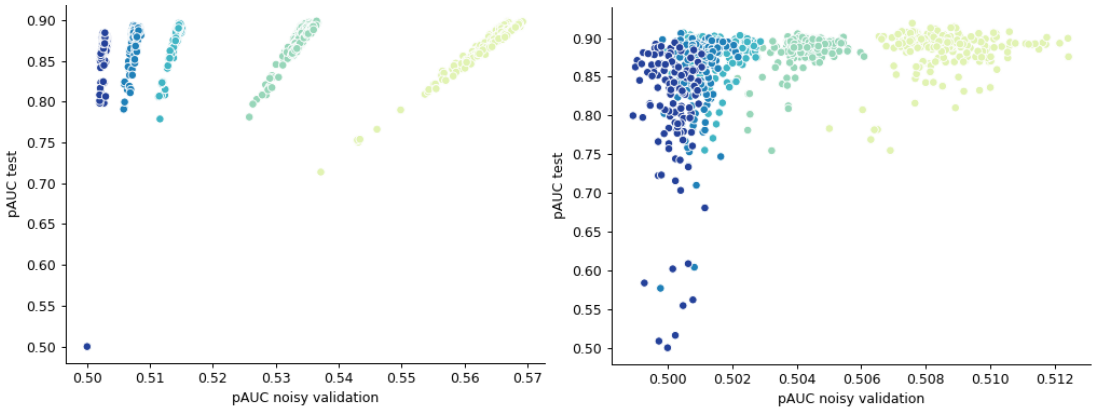
model predictions were impacted and this effect is accentuated with the increase of noise. In RF, the model is less confident at predicting both of the classes, and this result is consistent between all three random seeds. However, for LGBM, the results are not consistent, and we only see these same results in some of the seeds. For both RF and LGBM, the predictions of the model trained in NAR1 affected the confidence of the model at predicting the positive class. At last, similarly to NCAR and NAR0, the not at random label noises also skewed the probability distribution towards one, which means that RF and LGBM became more uncertain in presence of such noises. The NNAR NL and NNAR NN target mostly the majority class (Table A.2) and this can explain why this noise affected probability distribution in a similar way as NCAR and NAR0.

4.2 Evaluation on noisy validation

In this analysis, we looked into the Pearson's correlation between the pAUC in the noisy validation set and the pAUC in the test set. The goal was to understand whether well performing models in the noisy validation set are also well performing in the test set. In order to isolate which changes in the pAUC were caused by label noise, we also computed the results evaluated in the original validation for the same models.

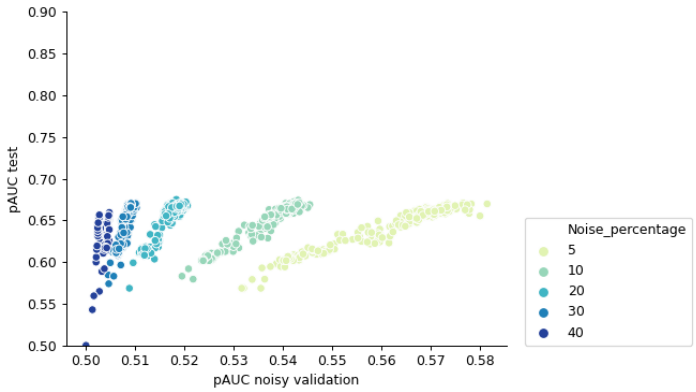
Table 4.2: Pearson correlation between original or noisy validation set and the test set.

			Test set		
Model	Label noise	Validation set	Banksim	CC	IEEE
RF	NCAR	Original	1.000	0.998	0.998
		Noisy	0.605	0.645	0.647
	NAR0	Original	1.000	0.999	0.998
		Noisy	0.660	0.692	0.746
	NAR1	Original	1.000	0.998	0.993
		Noisy	0.990	0.993	0.978
	NNAR NL	Original	0.998	0.781	0.988
		Noisy	-0.131	0.397	0.453
	NNAR NN	Original	0.998	0.917	0.997
		Noisy	0.172	0.387	0.840
LGBM	NCAR	Original	0.739	0.844	0.883
		Noisy	0.093	0.297	0.387
	NAR0	Original	0.721	0.848	0.799
		Noisy	-0.040	0.297	0.201
	NAR1	Original	0.797	0.973	0.824
		Noisy	0.784	0.971	0.730
	NNAR NL	Original	0.883	0.375	0.906
		Noisy	-0.150	-0.385	0.180
	NNAR NN	Original	0.896	0.868	0.954
		Noisy	0.030	0.550	0.603



(a) Banksim

(b) CC



(c) IEEE

Figure 4.5: Correlation between the noisy validation and the test pAUC for NCAR. Similarly to NCAR, the pAUC correlation also decreased with label noise percentage for NAR0 and NNAR.

In the end, we observed that the correlation decreased with the increase of label noise for NCAR, NAR0, and NNAR scenarios (Figure 4.5). Out of these noise types, the NNAR was the type of label noise that had the lowest pAUC correlations (Table 4.2). For this scenario, the model selection and evaluation were seriously compromised.

In addition, comparing between the pAUC correlations of RF and LGBM, we observed that the pAUC in the noisy validation were less correlated to the test set than in RF case (Table 4.2). Therefore, the evaluation for LGBM is the most affected by label noise. The correlations between the original validation and test sets also drop from RF to LGBM, which indicates that LGBM overfits more than RF. This is in line with the previous literature, which claimed that boosting algorithms are more prone to overfit the training set.

The only exception was for NAR1 scenario. Since this noise type only influences a small percentage of labels, it did not compromise the evaluation of the models. For this reason, there was a high pAUC score correlation between the noisy validation and test sets (all correlations higher than 0.73, as can be seen in Table 4.2), and the model selection was not affected.

4.3 Hyperparameter importance

To measure which hyperparameters contributed the most for the model performance, we iterated 100 different sets of parameters for both RF and LGBM (ranges and description are present in Table 3.3). Then, each of these trained models were evaluated in the test set in order to analyse how the performance varies depending on the parameters' values.

For RF, the variance of model performance was high for NCAR and NAR0 (Table 4.3), indicating that it is relevant to do a proper fine-tuning of the parameters and model selection while building a model in order to achieve good results.

Extending on the previous results about model performance for the top model for both algorithms in the presence of NAR1, the results for all the hyperparameter search trials show that the variance of the performance was low, and so, the models consistently reached good performance for this type of label noise (Table 4.3, and Figure 4.6). However, there was one exceptional case where the previous statement was not verified, which was for LGBM in CC (Figure 4.3).

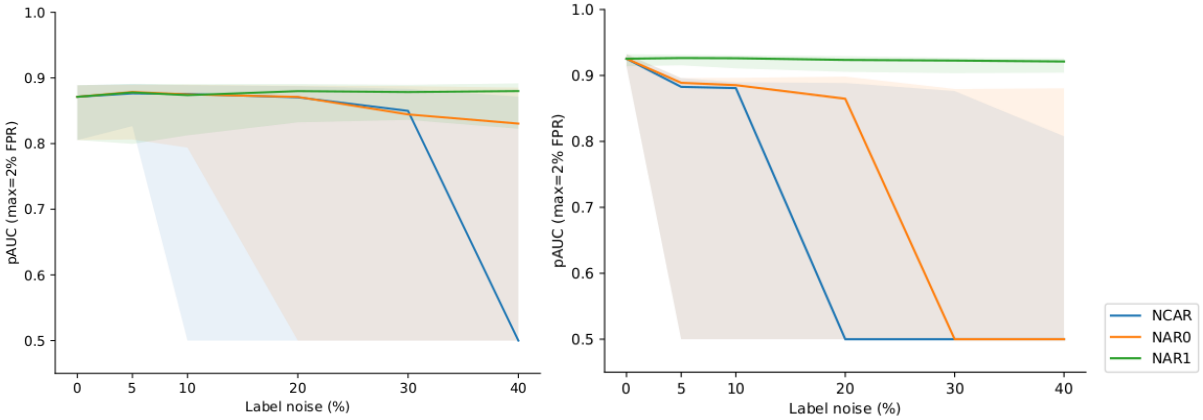
In the NNAR case, RF had little variance in pAUC values (Table 4.3). This shows that, independently of the choice of model parameters, most models performed poorly for NNAR scenario.

Regarding LGBM, we observed that the variance of model performance was small for all types of random and not at random label noise (Figure 4.3), with the only exception for NAR1 in the CC dataset. Thus, for most LGBM cases, we do not need to focus on a specific set of parameters in order to achieve a good model.

Since we wanted to identify the parameters that contribute the most for model performance, we evaluated how the parameter impacted the variance of the performance. For this, we used fANOVA and the most important parameter is the one that is responsible for the largest fraction of the variance and requires the correct tuning in order to obtain good performance. The NCAR and NAR0 had similar results in terms of parameter importance, and the parameters that contributed the most for the performance variance were the CCP alpha and class weight (Figure 4.7).

Table 4.3: Variance of performance per model and dataset and noise type

Model	Dataset	Noise type	Performance variance
RF	Banksim	NCAR	**0.033**
		NAR0	**0.032**
		NAR1	0.010
		NNAR NL	0.001
		NNAR NN	0.003
	CC	NCAR	**0.035**
		NAR0	**0.037**
		NAR1	0.016
		NNAR NL	0.000
		NNAR NN	0.000
	IEEE	NCAR	0.005
		NAR0	0.005
		NAR1	0.004
		NNAR NL	0.001
		NNAR NN	0.003
LGBM	Banksim	NCAR	0.003
		NAR0	0.002
		NAR1	0.011
		NNAR NL	0.006
		NNAR NN	0.004
	CC	NCAR	0.015
		NAR0	0.007
		NAR1	**0.040**
		NNAR NL	0.001
		NNAR NN	0.002
	IEEE	NCAR	0.001
		NAR0	0.000
		NAR1	0.001
		NNAR NL	0.001
		NNAR NN	0.004



(a) Banksim **(b)** CC

Figure 4.6: RF's pAUC score for the 100 sets of hyperparameters tested. The lines represent the median performance for the three random seeds (300 models). The filled areas show the interval between 25% and 75% quantiles for performance distribution.

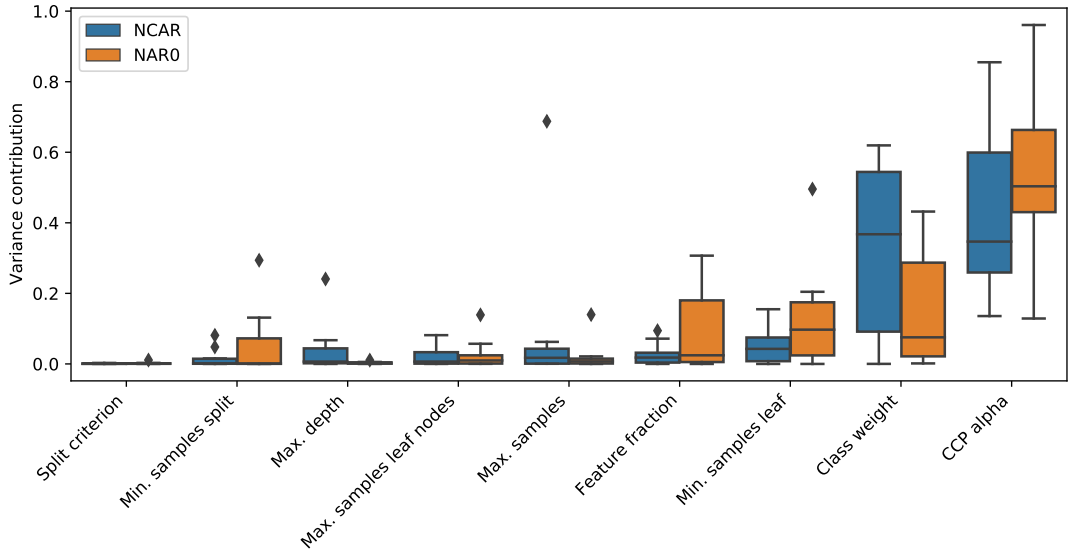
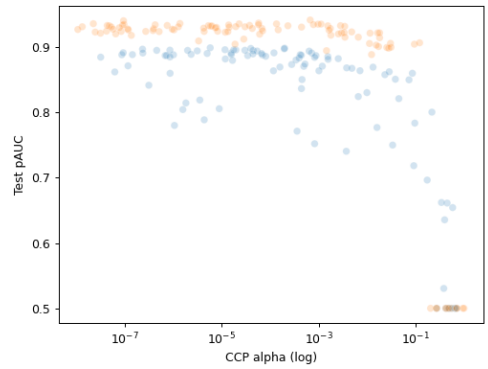
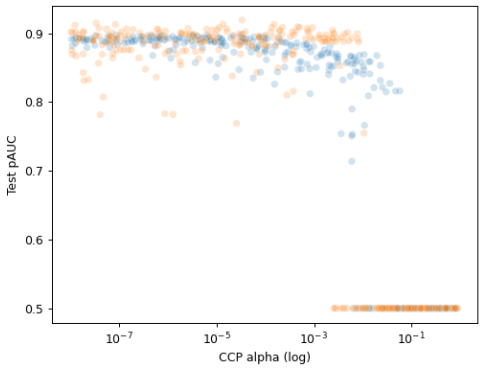


Figure 4.7: Variance contribution in the test set for the RF parameters in NCAR and NAR0 for Banksim and CC datasets. Ranges of parameters used are presented in Table 3.3

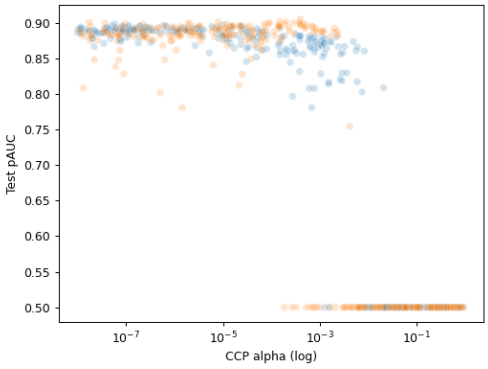
.



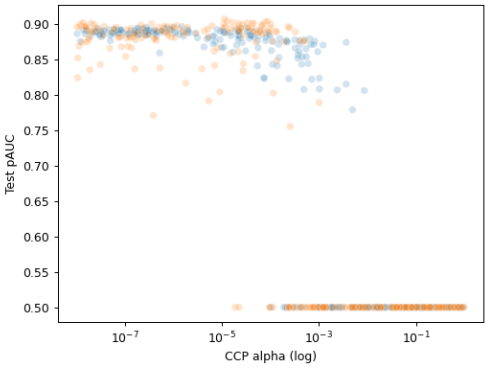
(a) 0%



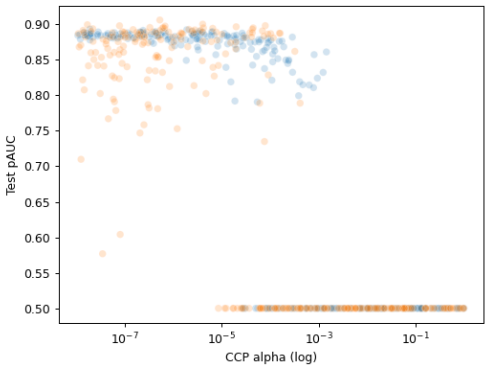
(b) 5%



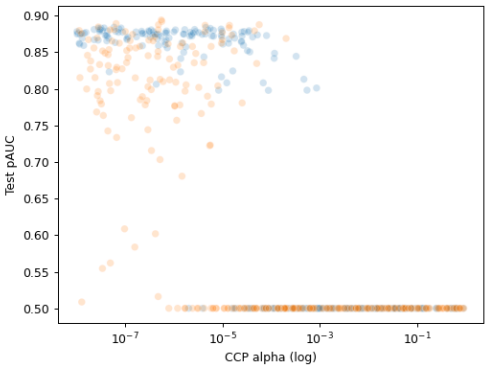
(c) 10%



(d) 20%



(e) 30%



(f) 40%

Figure 4.8: RF's pAUC score for the different values of CCP alpha in NCAR scenario. Blue data points correspond to Banskim dataset, and the orange data points to CC dataset.

As shown in Figure 4.10, as the noise percentage increases, lower values of CCP lead to better model performance. One possible explanation for these results is that performing pruning when in the presence of higher levels of label noise can lead to the removal of branches that are based on safe information [18].

The class weight was the second most important parameter, however, we could not find any clear patterns regarding the range of values that should be targeted to achieve good model performance.

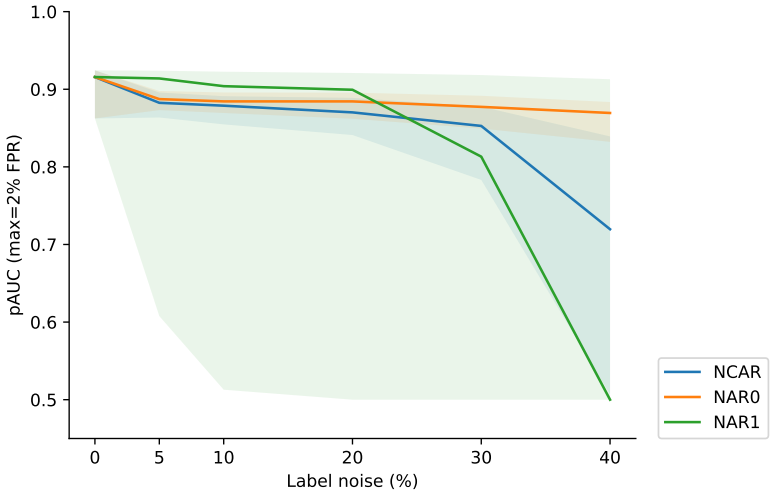
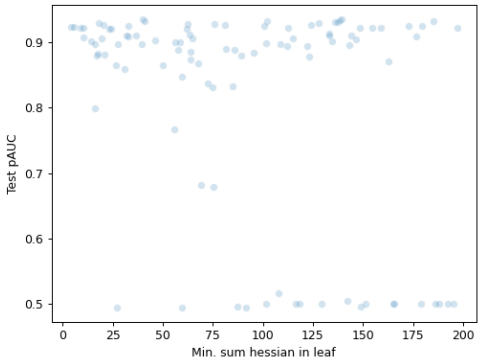


Figure 4.9: LGBM's pAUC score for the 100 sets of hyperparameters tested in NCAR and NAR scenarios for the CC dataset. The lines represent the median performance for the three random seeds (300 models). The filled areas show the interval between 25% and 75% quantiles for performance distribution.

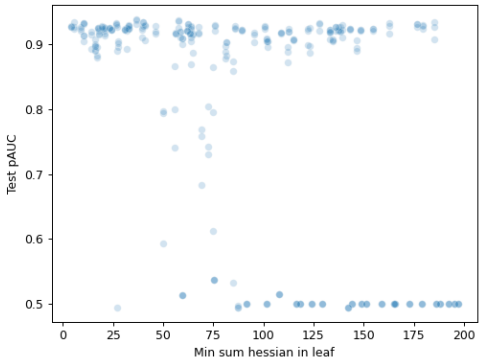
As mentioned before, the model performance for LGBM models was low for all types of label noises, apart from NAR1 in CC dataset (Figure 4.9 and Table 4.3). In this case, we analysed the most important parameter, and between all parameters, the minimum sum hessian in leaf was the one that contributed the most for the variance of model performance. For the baseline and low levels of artificial label noise, it is possible to achieve a good performing model for high ranges of the parameter minimum sum hessian in leaf. However, as we increase the level of noise, we could only achieve good models for a smaller values of the parameter that considers the minimum sum hessian in leaf. This parameter constrains the number of splits that are done in a LGBM model, and thus influencing the tree size, as a split only occurs if the minimum sum hessian for the leaf node is smaller than the value set for the parameter.

For both RF and LGBM, the most important parameters affect the size and the trees complexity. For both cases, the models that have better performance tend to be the ones that are bigger and more complex, as observed for NCAR and NAR0, and for

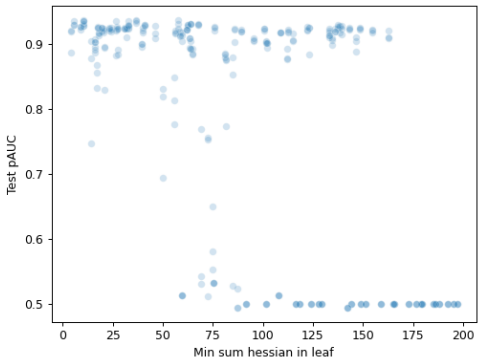
NAR1 in LGBM model. However, in LGBM, the minimum sum hessian in leaf is not always relevant to obtain good results, as it only affected the performance of CC.



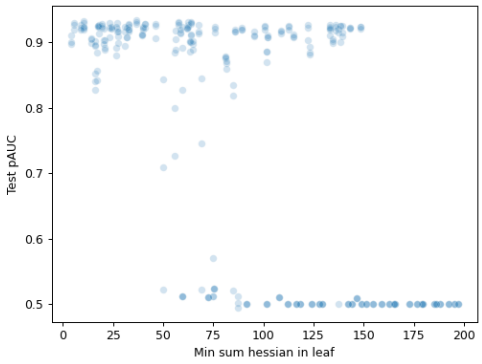
(a) 0%



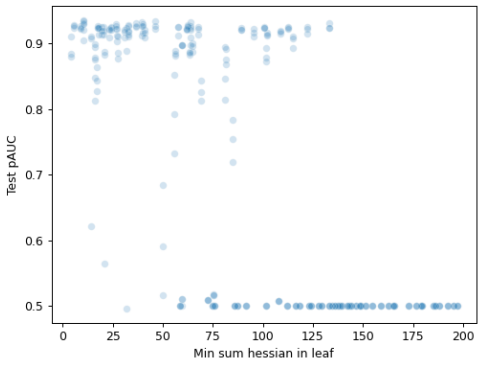
(b) 5%



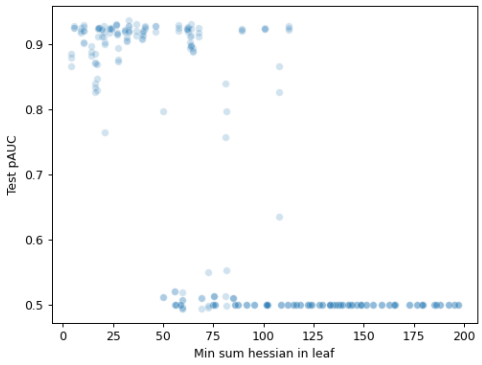
(c) 10%



(d) 20%



(e) 30%



(f) 40%

Figure 4.10: LightGBM's pAUC score for the different values of minimum sum hessian in leaf in NAR1 scenario in CC dataset.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This project was the first attempt to measure the impact of different types of label noise in model performance for fraud detection use-case.

We studied the impact on model training and hyperparameter tuning caused by the addition of different types of artificial label noise in tree-based algorithms. In our experiments, both Random Forest (RF) and LightGBM (LGBM) are robust to Noise Completely at Random (NCAR) and Noise at Random (NAR), but fail in the presence of Noise Not at Random (NNAR). In NCAR and NAR case, RF seems to be more impacted than LGBM since the 100 tested models had a higher variance in performance. Even so, a good performing model can still be selected in the noisy validation set. In the case of LGBM, the variance of performance is very small for all types of label noise, indicating that changes in the parameters' values do not affect model performance in a noisy setting.

NNAR models trained with different sets of hyperparameters in these types of noise had poor performances in the unchanged test set, thus none of the parameters seemed able to overcome label noise. Alternative techniques to overcome the detrimental effects showed by NNAR are extremely necessary, as this type of noise is possibly the most realistic. This is because instances close to the borderline have higher chances of being misjudged by fraud analysts. Preliminary results of fraud analysts review agreement showed that the level of agreement between fraud analysts is low. For this reason, it is expected that fraud marking has more disagreement in examples that are harder to classify between different analysts. One possible reason that can lead to different labels between analysts is the number of years of experience in the field. However, how the level of disagreement is related to the distance to the decision

boundary is still unknown.

In Feedzai's use-case, models are optimised to achieve high levels of recall at small FPR, generally 1%. Thus, in our experiments, we used pAUC for a range between 0 and 2%. We observed that the introduction of label noise in the validation set had a detrimental effect on pAUC, leading to an underestimation of the real model performance. Ultimately, the pAUC in the noisy validation becomes less correlated to the pAUC in the test set as we increase the label noise, especially in NNAR. Therefore, label noise not only impacts the estimation of model performance but also impacts model selection. This is particularly more relevant in RF since the models generated had higher variance of pAUC for the different sets of parameters, which means that RF's parameters are more sensitive to label noise than LGBM.

Models evaluated in noisy setups can bias the model selection towards models that fitted noisy labels instead of models that are actually more robust to noise. Exploratory analysis on alternative metrics that do not directly depend on the fraud rate (e.g. precision, brier score and F1-score) also showed that these metrics were not effective for assessing model performance in the presence of such label noises.

Contrarily to previous studies which state that boosting algorithms are more sensitive to label noise than bagging to random label noise [39], in our study RF was as robust to random as LGBM. In NNAR, both models decayed in performance at similar rates with the increase of noise. Thus, there was not a clear distinction between the two algorithms.

Finally, our observations on the hyperparameter importance for both algorithms show that, for RF models, CCP alpha was the parameter that contributed the most for the performance variance, and results imply that, for greater percentages of label noise, lower values of the parameter lead to better results. As for LGBM, although some results attribute greater importance to the min sum hessian in leaf parameter, where lower values lead to better results, this can not be generalized. Both of the parameters referred before affect the size of the trees of their respective algorithms, and build bigger and more complex trees for higher levels of label noise.

5.2 Future Work

One of the biggest limitations of studying label noise in real applications is not having a feasible way of assessing which noise types are present in real-life datasets and at which rates they appears. We tried to mimic different types of noise, including random and not at random label noise, but it is still unclear for us how close these scenarios are to reality. We expect that label noise not at random will be closer to real label noise because we expect that fraud analysts will have more difficulty in classifying the data points in the classification boundary and consequently, these points will be potentially noisier than the remaining data.

The robustness of the model was assessed for the imbalanced binary classification case, and we cannot extend these results to different set-ups, for instance, balanced datasets and multiclass problems.

After identifying the detrimental effects of label noise, the next steps of the project are to test different methodologies to identify and overcome not at random label noise. There are a variety of methods that increase model robustness, including robust log losses that target the model training or the use of cleansing methods. The latter can be done before training the model and could solve the different issues presented here. By applying these techniques in the training set, the model could be built with higher robustness in addition to the application on the validation set that would improve our model performance estimation and parameters' choice.

However, for NNAR, standard filter methodologies, such as ensemble or neighbourhood based filters [19], do not identify noise as accurately as for NAR. Thus, it would be interesting to benchmark cleansing techniques and apply them to real-life datasets at Feedzai. We expect that training and tuning the model in cleansed sets will have a performance gain when compared with the current best model trained in noisy data. This could potentially help deploying models in production that are robust to label noise and better evaluate model performance.

Bibliography

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. "Optuna: A Next-generation Hyperparameter Optimization Framework." In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019), pp. 2623–2631. arXiv: 1907.10902. URL: http://arxiv.org/abs/1907.10902.

[2] C. Bergmeir and J. M. Benítez. "On the use of cross-validation for time series predictor evaluation." In: *Information Sciences* 191 (2012), pp. 192–213. ISSN: 00200255. DOI: 10.1016/j.ins.2011.12.028.

[3] R. J. Bolton and D. J. Hand. *Statistical Fraud Detection: A Review*. DOI: 10.2307/3182781. URL: https://www.jstor.org/stable/3182781.

[4] L. Breiman. "Randomizing outputs to increase prediction accuracy." In: *Machine Learning* 40.3 (2000), pp. 229–242. ISSN: 08856125. DOI: 10.1023/A:1007682208299.

[5] L. Breiman. "Random forests." In: *Machine Learning* 45.1 (2001), pp. 5–32. ISSN: 08856125. DOI: 10.1023/A:1010933404324.

[6] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification And Regression Trees*. Wadsworth & Brooks/Cole Advanced Books & Software, 1984.

[7] C. E. Brodley and M. A. Friedl. "Identifying Mislabeled Training Data." In: *Journal of Artificial Intelligence Research* 11 (1999), pp. 131–167. ISSN: 1076-9757. DOI: 10.1613/jair.606. arXiv: 1106.0219. URL: http://arxiv.org/abs/1106.0219.

[8] R. S. Chhikara and J. McKeon. "Linear Discriminant Analysis with Misallocation in Training Samples." In: *Journal of the American Statistical Association* 79.388 (1984), p. 899. ISSN: 01621459. DOI: 10.2307/2288722. URL: https://www.jstor.org/stable/2288722?origin=crossref.

[9] *Credit Card Fraud Detection | Kaggle*. URL: https://www.kaggle.com/mlg-ulb/creditcardfraud (visited on 04/29/2020).

[10] T. G. Dietterich. "Experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization." In: *Machine Learning* 40.2 (2000), pp. 139–157. ISSN: 08856125. DOI: 10.1023/A:1007607513941.

[11] C. Elkan. "The Foundations of Cost-Sensitive Learning." In: *In Proceedings of the Seventeeth International Joint Conference on Artificial Intelligence* (2001), pp. 973–978. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.514.

[12] M. Feurer, J. T. Springenberg, and F. Hutter. "Initializing Bayesian hyperparameter optimization via meta-learning." In: *Proceedings of the National Conference on Artificial Intelligence*. Vol. 2. 2015, pp. 1128–1135. ISBN: 9781577357001.

[13] A. Folleco, T. M. Khoshgoftaar, J. Van Hulse, and L. Bullard. "Identifying learners robust to low quality data." In: *2008 IEEE International Conference on Information Reuse and Integration*. IEEE, 2008, pp. 190–195. ISBN: 978-1-4244-2659-1. DOI: 10.1109/IRI.2008.4583028. URL: http://ieeexplore.ieee.org/document/4583028/.

[14] *Fraud Losses in 2019 Topped $1.9B, FTC Reports*. URL: https://www.aarp.org/money/scams-fraud/info-2020/ftc-fraud-complaints-rise.html{\#}:{~}:text=0verallin2019{\%}2Cnearly{\%}24667,fora28percentjump (visited on 07/27/2020).

[15] B. Frénay, G. Doquire, and M. Verleysen. "Estimating mutual information for feature selection in the presence of label noise." In: *Computational Statistics and Data Analysis* 71 (2014), pp. 832–848. ISSN: 01679473. DOI: 10.1016/j.csda.2013.05.001.

[16] B. Frénay and M. Verleysen. "Classification in the presence of label noise: A survey." In: *IEEE Transactions on Neural Networks and Learning Systems* 25.5 (2014), pp. 845–869. ISSN: 21622388. DOI: 10.1109/TNNLS.2013.2292894.

[17] Y. Freund and R. E. Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting." In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139. ISSN: 00220000. DOI: 10.1006/jcss.1997.1504.

[18] L. P. Garcia, A. C. de Carvalho, and A. C. Lorena. "Effect of label noise in the complexity of classification problems." In: *Neurocomputing* 160 (2015), pp. 108–119. ISSN: 09252312. DOI: 10.1016/j.neucom.2014.10.085. URL: https://linkinghub.elsevier.com/retrieve/pii/S0925231215001241.

[19] L. P. Garcia, J. Lehmann, A. C. de Carvalho, and A. C. Lorena. "New label noise injection methods for the evaluation of noise filters." In: *Knowledge-Based Systems* 163 (2019), pp. 693–704. ISSN: 09507051. DOI: 10.1016/j.knosys.2018.09.031. URL: https://linkinghub.elsevier.com/retrieve/pii/S0950705118304829.

[20] A. Ghosh, N. Manwani, and P. S. Sastry. "On the Robustness of Decision Tree Learning under Label Noise." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10234 LNAI (2016), pp. 685–697. ISSN: 16113349. DOI: 10.1007/978-3-319-57454-7_53. arXiv: 1605.06296. URL: http://link.springer.com/10.1007/978-3-319-57454-7_53.

[21] *GitHub - microsoft/LightGBM: A fast, distributed, high performance gradient boosting (GBT, GBDT, GBRT, GBM or MART) framework based on decision tree algorithms, used for ranking, classification and many other machine learning tasks.* URL: https://github.com/microsoft/LightGBM (visited on 06/29/2020).

[22] *GitHub - mwaskom/seaborn: Statistical data visualization using matplotlib*. URL: https://github.com/mwaskom/seaborn (visited on 06/26/2020).

[23] *GitHub - optuna/optuna: A hyperparameter optimization framework*. URL: https://github.com/optuna/optuna (visited on 08/15/2020).

[24] *GitHub - wwwjk366/gower: Python package for Gower distance*. URL: https://github.com/wwwjk366/gower (visited on 06/29/2020).

[25] A. Gómez-Ríos, J. Luengo, and F. Herrera. "A study on the noise label influence in boosting algorithms: Adaboost, GBM and XGBoost." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10334 LNCS. Springer Verlag, 2017, pp. 268–280. ISBN: 9783319596495. DOI: 10.1007/978-3-319-59650-1_23.

[26] J. C. Gower. "A General Coefficient of Similarity and Some of Its Properties." In: *Biometrics* 27.4 (1971), p. 857. ISSN: 0006341X. DOI: 10.2307/2528823. URL: https://www.jstor.org/stable/2528823?origin=crossref.

[27] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY: Springer New York, 2009, pp. 1–656. ISBN: 978-0-387-84857-0. DOI: 10.1007/978-0-387-84858-7. arXiv: arXiv:1011.1669v3. URL: http://link.springer.com/10.1007/978-0-387-84858-7.

[28] F. Hutter, H. Hoos, and K. Leyton-Brown. *An Efficient Approach for Assessing Hyperparameter Importance*. Tech. rep. URL: http://proceedings.mlr.press/v32/hutter14.html.

[29] *IEEE-CIS Fraud Detection | Kaggle*. URL: https://www.kaggle.com/c/ieee-fraud-detection (visited on 04/29/2020).

[30] P. G. Ipeirotis, F. Provost, and J. Wang. "Quality management on Amazon Mechanical Turk." In: *Workshop Proceedings - Human Computation Workshop 2010, HCOMP2010*. New York, New York, USA: ACM Press, 2010, pp. 64–67. ISBN: 9781450302227. DOI: 10.1145/1837885.1837906. URL: http://portal.acm.org/citation.cfm?doid=1837885.1837906.

[31] *Kaggle: Your Home for Data Science*. URL: https://www.kaggle.com/ (visited on 03/31/2020).

[32] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. *LightGBM: A Highly Efficient Gradient Boosting Decision Tree*. Tech. rep. 2017, pp. 3146–3154. URL: https://github.com/Microsoft/LightGBM..

[33] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. "Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets." In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017* (2016). arXiv: 1605.07079. URL: http://arxiv.org/abs/1605.07079.

[34] C. P. Lam and D. G. Stork. "Evaluating classifiers by means of test data with noisy labels." In: *IJCAI International Joint Conference on Artificial Intelligence*. 2003, pp. 513–518. DOI: 10.5555/1630659.1630735.

[35] E. A. Lopez-Rojas and S. Axelsson. "Banksim: A bank payments simulator for fraud detection research." In: *26th European Modeling and Simulation Symposium, EMSS 2014*. 2014, pp. 144–152. ISBN: 9788897999324.

[36] G. Martínez-Muñoz, A. Sánchez-Martínez, D. Hernández-Lobato, and A. Suárez. "Building ensembles of neural networks with class-switching." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 4131 LNCS - I. Springer Verlag, 2006, pp. 178–187. ISBN: 3540386254. DOI: 10.1007/11840817_19.

[37] G. Martínez-Muñoz, A. Sánchez-Martínez, D. Hernández-Lobato, and A. Suárez. "Class-switching neural network ensembles." In: *Neurocomputing*. Vol. 71. 13-15. Elsevier, 2008, pp. 2521–2528. DOI: 10.1016/j.neucom.2007.11.041.

[38] G. Martínez-Muñoz and A. Suárez. "Switching class labels to generate classification ensembles." In: *Pattern Recognition* 38.10 (2005), pp. 1483–1494. ISSN: 00313203. DOI: 10.1016/j.patcog.2005.02.020.

[39] P. Melville, N. Shah, L. Mihalkova, and R. J. Mooney. "Experiments on Ensembles with Missing and Noisy Data." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 3077. 2004, pp. 293–302. DOI: 10.1007/978-3-540-25966-4_29. URL: https://link.springer.com/chapter/10.1007/978-3-540-25966-4_29.

[40] D. F. Nettleton, A. Orriols-Puig, and A. Fornells. "A study of the effect of different types of noise on the precision of supervised learning techniques." In: *Artificial Intelligence Review* 33.3-4 (2010), pp. 275–306. ISSN: 02692821. DOI: 10.1007/s10462-010-9156-z.

[41] *Press - Feedzai*. URL: https://feedzai.com/press/ (visited on 08/16/2020).

[42] J. R. Quinlan. *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN: 1558602402.

[43] K. V. Rashmi and R. Gilad-Bachrach. "DART: Dropouts meet Multiple Additive Regression Trees." In: *Journal of Machine Learning Research* 38 (2015), pp. 489–497. arXiv: 1505.01866. URL: http://arxiv.org/abs/1505.01866.

[44] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. "Learning from crowds." In: *Journal of Machine Learning Research* 11 (2010), pp. 1297–1322. ISSN: 15324435. URL: http://www.jmlr.org/papers/v11/raykar10a.

[46] J. A. Sáez, J. Luengo, and F. Herrera. "Evaluating the classifier behavior with noisy data considering performance and robustness: The Equalized Loss of Accuracy measure." In: *Neurocomputing* 176 (2016), pp. 26–35. ISSN: 09252312. DOI: 10.1016/j.neucom.2014.11.086. URL: https://linkinghub.elsevier.com/retrieve/pii/S0925231215005500.

[47] V. S. Sheng, R. Tada, and A. Atla. "An Empirical Study of Class Noise Impacts on Supervised Learning Algorithms and Measures." In: *Proceedings of the International Conference on Data Mining (DMIN)* (2011). URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.217.7650.

[48] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. "Cheap and fast - But is it good? Evaluating non-expert annotations for natural language tasks." In: *EMNLP 2008 - 2008 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference: A Meeting of SIGDAT, a Special Interest Group of the ACL*. 2008, pp. 254–263.

[49] *Synthetic data from a financial payment system | Kaggle*. URL: https://www.kaggle.com/ntnu-testimon/banksim1 (visited on 04/29/2020).

[50] C. M. Teng. "Correcting noisy data." In: *Proc 16th International Conf on Machine Learning* (1999), pp. 239–248. ISSN: - 0952-3871. DOI: 10.1007/978-3-319-32034-2_46_A.

[51] Tin Kam Ho and M. Basu. "Complexity measures of supervised classification problems." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.3 (2002), pp. 289–300. ISSN: 01628828. DOI: 10.1109/34.990132. URL: http://ieeexplore.ieee.org/document/990132/.

[52] D. P. Williams. "Label alteration to improve underwater mine classification." In: *IEEE Geoscience and Remote Sensing Letters* 8.3 (2011), pp. 488–492. ISSN: 1545598X. DOI: 10.1109/LGRS.2010.2088106.

[53] L. Yang and A. Shami. "On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice." In: *Neurocomputing* (2020). DOI: 10.1016/j.neucom.2020.07.061. arXiv: 2007.15745. URL: http://arxiv.org/abs/2007.15745.

[54] M.-C. Yuen, I. King, and K.-S. Leung. "A Survey of Crowdsourcing Systems." In: 2012, pp. 766–773. DOI: 10.1109/passat/socialcom.2011.203.

[55] X. Zhu and X. Wu. "Class Noise vs. Attribute Noise: A Quantitative Study." In: *Artificial Intelligence Review* 22.3 (2004), pp. 177–210. ISSN: 0269-2821. DOI: 10.1007/s10462-004-0751-8.

[56] X. Zhu, X. Wu, and Q. Chen. "Eliminating Class Noise in Large Datasets." In: *Proceedings, Twentieth International Conference on Machine Learning*. Vol. 2. 2003, pp. 920–927. ISBN: 1577351894.

Appendix

Appendix 1

Table A.1: SVM performance results for nonlinearwise label noise.

Dataset	Recall	Precision	Alert rate
Banksim	99.46%	13.00%	9.56%
CC	90.22%	8.79%	1.81%
IEEE	11.30%	56.74%	20.16%

Table A.2: Characteristics of the artificial label noise in the training set.

Dataset	Type of label noise	Noise level	Seed	Changes 0 -> 1	Changes 1 -> 0	Fraud (%)
Banksim	NCAR	5%	1	20546	266	6.135%
Banksim	NCAR	5%	2	20574	238	6.148%
Banksim	NCAR	5%	3	20556	256	6.140%
Banksim	NCAR	10%	1	41083	542	11.003%
Banksim	NCAR	10%	2	41109	516	11.015%
Banksim	NCAR	10%	3	41131	494	11.026%
Banksim	NCAR	20%	1	82206	1044	20.761%
Banksim	NCAR	20%	2	82196	1054	20.757%
Banksim	NCAR	20%	3	82249	1001	20.782%
Banksim	NCAR	30%	1	123304	1571	30.508%
Banksim	NCAR	30%	2	123257	1618	30.486%
Banksim	NCAR	30%	3	123357	1518	30.534%
Banksim	NCAR	40%	1	164395	2105	40.252%

Banksim	NCAR	40%	2	164335	2165	40.223%
Banksim	NCAR	40%	3	164449	2051	40.277%
Banksim	NAR0	5%	1,2,3	20549	-	6.200%
Banksim	NAR0	10%	1,2,3	41099	-	11.137%
Banksim	NAR0	20%	1,2,3	82198	-	21.010%
Banksim	NAR0	30%	1,2,3	123297	-	30.884%
Banksim	NAR0	40%	1,2,3	164397	-	40.758%
Banksim	NAR1	5%	1,2,3	-	262	1.200%
Banksim	NAR1	10%	1,2,3	-	525	1.137%
Banksim	NAR1	20%	1,2,3	-	1051	1.010%
Banksim	NAR1	30%	1,2,3	-	1577	0.088%
Banksim	NAR1	40%	1,2,3	-	2102	0.076%
Banksim	NNAR NL	5%	-	20576	236	6.149%
Banksim	NNAR NL	10%	-	40949	676	10.938%
Banksim	NNAR NL	20%	-	81198	2052	20.277%
Banksim	NNAR NL	30%	-	122064	2811	29.912%
Banksim	NNAR NL	40%	-	162914	3586	39.540%
Banksim	NNAR NN	5%	-	17163	3649	4.510%
Banksim	NNAR NN	10%	-	36453	5171	8.778%
Banksim	NNAR NN	20%	-	78014	5236	18.747%
Banksim	NNAR NN	30%	-	119638	5237	28.747%
Banksim	NNAR NN	40%	-	161259	5241	38.745%
CC	NCAR	5%	1	9909	21	5.163%
CC	NCAR	5%	2	9913	17	5.167%
CC	NCAR	5%	3	9917	13	5.170%
CC	NCAR	10%	1	19817	43	10.141%
CC	NCAR	10%	2	19825	35	10.149%
CC	NCAR	10%	3	19828	32	10.152%
CC	NCAR	20%	1	39632	89	20.094%
CC	NCAR	20%	2	39646	75	20.108%
CC	NCAR	20%	3	39661	60	20.124%
CC	NCAR	30%	1	59467	115	30.068%
CC	NCAR	30%	2	59468	114	30.069%
CC	NCAR	30%	3	59486	96	30.087%
CC	NCAR	40%	1	79319	124	40.059%
CC	NCAR	40%	2	79298	145	40.038%

CC	NCAR	40%	3	79319	124	40.059%
CC	NAR0	5%	1,2,3	9912	-	5.175%
CC	NAR0	10%	1,2,3	19824	-	10.166%
CC	NAR0	20%	1,2,3	39648	-	20.147%
CC	NAR0	30%	1,2,3	59472	-	30.129%
CC	NAR0	40%	1,2,3	79296	-	40.110%
CC	NAR1	5%	1,2,3	-	18	0.175%
CC	NAR1	10%	1,2,3	-	36	0.166%
CC	NAR1	20%	1,2,3	-	73	0.148%
CC	NAR1	30%	1,2,3	-	109	0.129%
CC	NAR1	40%	1,2,3	-	146	0.111%
CC	NNAR NL	5%	-	9908	22	5.162%
CC	NNAR NL	10%	-	19860	36	10.148%
CC	NNAR NL	20%	-	39655	66	20.118%
CC	NNAR NL	30%	-	59508	74	30.119%
CC	NNAR NL	40%	-	79366	77	40.107%
CC	NNAR NN	5%	-	9727	203	4.880%
CC	NNAR NN	10%	-	19630	230	9.952%
CC	NNAR NN	20%	-	39454	269	19.915%
CC	NNAR NN	30%	-	59279	303	29.879%
CC	NNAR NN	40%	-	79113	330	39.852%
IEEE	NCAR	5%	1	19925	743	8.157%
IEEE	NCAR	5%	2	19931	737	8.160%
IEEE	NCAR	5%	3	19948	720	8.168%
IEEE	NCAR	10%	1	39931	1406	12.836%
IEEE	NCAR	10%	2	39873	1464	12.808%
IEEE	NCAR	10%	3	39892	1445	12.818%
IEEE	NCAR	20%	1	79784	2891	22.118%
IEEE	NCAR	20%	2	79762	2913	22.107%
IEEE	NCAR	20%	3	79790	2885	22.121%
IEEE	NCAR	30%	1	119673	4340	31.417%
IEEE	NCAR	30%	2	119644	4369	31.403%
IEEE	NCAR	30%	3	119619	4394	31.391%
IEEE	NCAR	40%	1	159502	5849	40.687%
IEEE	NCAR	40%	2	159458	5893	40.666%
IEEE	NCAR	40%	3	159553	5798	40.712%

IEEE	NAR0	5%	1,2,3	19942	-	8.341%
IEEE	NAR0	10%	1,2,3	39884	-	13.165%
IEEE	NAR0	20%	1,2,3	79768	-	22.814%
IEEE	NAR0	30%	1,2,3	119652	-	32.462%
IEEE	NAR0	40%	1,2,3	159536	-	42.110%
IEEE	NAR1	5%	1,2,3	-	726	3.341%
IEEE	NAR1	10%	1,2,3	-	1453	3.165%
IEEE	NAR1	20%	1,2,3	-	2907	2.814%
IEEE	NAR1	30%	1,2,3	-	4361	2.462%
IEEE	NAR1	40%	1,2,3	-	5815	2.110%
IEEE	NNAR NL	5%	-	19302	1366	7.856%
IEEE	NNAR NL	10%	-	39038	2299	12.404%
IEEE	NNAR NL	20%	-	78466	4209	21.480%
IEEE	NNAR NL	30%	-	118355	5658	30.779%
IEEE	NNAR NL	40%	-	158568	6783	40.235%
IEEE	NNAR NN	5%	-	13822	6846	5.204%
IEEE	NNAR NN	10%	-	33432	7905	9.692%
IEEE	NNAR NN	20%	-	73753	8922	19.200%
IEEE	NNAR NN	30%	-	114226	9787	28.782%
IEEE	NNAR NN	40%	-	154777	10574	38.401%